



eHealth[®] SystemEDGE[™] User Guide

MN-SEAGTUGD-007
November 2006

This documentation (the "Documentation") and related computer software program (the "Software") (hereinafter collectively referred to as the "Product") is for the end user's informational purposes only and is subject to change or withdrawal by CA at any time.

This Product may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Product is confidential and proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of the Documentation for their own internal use, and may make one copy of the Software as reasonably required for back-up and disaster recovery purposes, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the provisions of the license for the Software are permitted to have access to such copies.

The right to print copies of the Documentation and to make a copy of the Software is limited to the period during which the license for the Product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to certify in writing to CA that all copies and partial copies of the Product have been returned to CA or destroyed.

EXCEPT AS OTHERWISE STATED IN THE APPLICABLE LICENSE AGREEMENT, TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS PRODUCT "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS PRODUCT, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

The use of this Product and any product referenced in the Documentation is governed by the end user's applicable license agreement.

The manufacturer of this Product is CA.

This Product is provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7013(c)(1)(ii), as applicable, or their successors.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Copyright © 2006 CA. All rights reserved.

Table of Contents

Preface	23
Audience	24
About This Guide	24
Revision Information	24
Documentation Conventions	25
Technical Support	26
Current SystemEDGE Product Information	26

Chapter 1	Introduction	27
	Introducing eHealth SystemEDGE	27
	Microsoft® Data Center Certification	27
	Using eHealth SystemEDGE	28
	Supported MIBs	28
	SystemEDGE Self-Monitoring Features	31
	Identifying Top Processes	33
	Tracking Assets	34
	Supporting Custom MIB Objects	34
	Supporting Windows Registry and Perfmon Extensions	34
	Specifying Corrective Actions	35
	SystemEDGE in Windows Clustered Environment	36
	Using eHealth AdvantEDGE View	37
	Using eHealth Service Availability	37
	Monitoring Voice and Call Quality	38
	Using the eHealth Application Insight Modules	39

Using eHealth with eHealth SystemEDGE.....	40
Using eHealth Live Health — Fault Manager with eHealth SystemEDGE	42
Guidelines for Using the SystemEDGE Agent.....	43
Simple Network Management Protocol	47
SNMP Message Types.....	48
SNMP Communities	48

Chapter 2

Installing the SystemEDGE Agent

51

Installing SystemEDGE on Solaris Systems.....	52
System Requirements	52
Adding the SystemEDGE Agent Package	52
Installing SystemEDGE in a Non-Default Directory for Solaris.....	58
Installing SystemEDGE on Windows Systems	59
System Requirements	59
Installing the Software with InstallShield	60
Installing the Software from the Command Line	74
Installing SystemEDGE on HP-UX Systems.....	75
System Requirements	75
Installing the Software	75
Installing SystemEDGE on Linux Systems	80
System Requirements	80
Installing the Software	81
Installing SystemEDGE on AIX Systems	82
System Requirements	82
Installing the Software	83
Installing SystemEDGE on Digital UNIX and Tru64 Systems. ...	84
System Requirements	84
Installing the Software	84
Reviewing the Configuration Files	86
Configuration Files for UNIX Systems	89
Configuration Files for Windows Systems	89
SystemEDGE Agent Directories and Files	90
Directories and Files for UNIX Systems	90
Directories and Files for Windows Systems	92
Directories and Files for All Platforms	94

Uninstalling the SystemEDGE Agent	98
Uninstalling SystemEDGE for Solaris Systems	98
Uninstalling SystemEDGE for Windows Systems	98
Uninstalling SystemEDGE for HP-UX Systems	101
Uninstalling SystemEDGE for Linux Systems	102
Uninstalling SystemEDGE for AIX Systems	102
Uninstalling SystemEDGE for Digital UNIX and Tru64 Systems	103

Chapter 3 Licensing the SystemEDGE Agent 105

Licensing the Agent	105
License Keys	106
Licensing Methods	107
Obtaining a License Key	108
Running the licenseeme Utility	109
Sending an E-mail Request for a License Key	110
Using the Web-Based Licensing Form	111
Using the licenseutil.pl Utility	112

Chapter 4 Configuring the SystemEDGE Agent 117

Interactions Between sysedge.cf and sysedge.mon	118
Configuring the Agent During the Installation Procedure	118
Before You Begin	118
Sample sysedge.cf File	119
Configuring System Information	128
Configuring Access Communities	128
Specifying the Access List	129
Default Settings	130
Configuring Trap Communities	130
Specifying a Trap Source	131
Configuring Authentication Failure Traps	132
Configuring Support for Who Table Information	132
Configuring Support for User and Group Information	133
Configuring Support for Remote Shell Capability	133
Configuring Alternative Syslog Facilities (UNIX Only)	134
Configuring Alternative Syslog Facilities (Windows Only)	134

Configuring Support for Agent Debugging.	135
Configuring Support for Floppy Status Checking	135
Configuring Support for Serial Port Status Checking	136
Configuring Support for Disk Probing	136
Configuring Support for Actions	137
Disabling Support for Remote File System Checking (UNIX Only)	138
Configuring Support for Threshold Monitoring	138
Configuring Support for Process Monitoring.	139
Monitoring Applications, Processes, and Services.	140
Monitoring Process Attributes.	140
Monitoring Windows Services.	140
Configuring Support for Process Group Monitoring.	141
Configuring Support for Log File Monitoring	141
Configuring Support for Windows Event Log Monitoring (Windows Only)	141
Configuring History Collection	142
Configuring User/Group Permissions for Subprograms (UNIX Only)	142
Configuring the SNMP Bind Address	143
Configuring Support for eHealth AIMs.	143
Configuring Support for Linux Free Memory	145
Recommendations for Configuring Security.	146
Using the SystemEDGE Control Panel for Windows.	147

Chapter 5 Starting the SystemEDGE Agent 149

Starting the Agent Manually	149
Starting SystemEDGE on Windows Systems	149
Starting SystemEDGE on UNIX Systems	150
Starting the Agent Automatically at System Boot.	151
Starting the Agent Automatically for Solaris Systems	152
Starting the Agent Automatically for Windows Systems.	153
Starting the Agent Automatically for HP-UX Systems	153
Starting the Agent Automatically for Linux Systems.	154
Starting the Agent Automatically for AIX Systems	154
Starting the Agent Automatically for Digital UNIX or Tru64 Systems.	155

	Logging Agent Operation Messages.....	155
	Logging Messages for UNIX	155
	Logging Messages for Windows	156
Chapter 6	Using the SystemEDGE Agent with Other SNMP Agents	157
	Supporting Multiple SNMP Agents.....	157
	Agent Multiplexing.....	158
	Monolithic Agents	159
	Using the SystemEDGE Agent with the Solstice Enterprise Agent	160
	Using the SystemEDGE Agent with the Microsoft Windows SNMP Agent	161
	Using the SystemEDGE Agent with the HP SNMP Agent.....	162
	Using the SystemEDGE Agent with the AIX SNMP Agent.....	163
	Using the SystemEDGE Agent with the Digital UNIX or Tru64 SNMP Agent.....	163
	Using the SystemEDGE Agent with the Compaq Insight Manager	164
Chapter 7	Systems Management MIB	165
	Host System Information	166
	Mounted Devices.....	167
	Monitoring File System Space.....	168
	Unmounting a Mounted Device.....	168
	Kernel Configuration	168
	Boot Configuration	169
	Streams Group.....	170
	User Information.....	172
	Group Information	173
	Process Information	174
	Changing the nice Value of a Process (UNIX only)	175
	Sending a Signal to a Process.....	176
	Who Table Information	176
	Remote Command Execution	177
	Executing a Remote Command	178

Kernel Performance Statistics	178
Interprocess Communication: Queues, Shared Memory, and Semaphores	180
Deleting an Interprocess Communication	181
Message Buffer Allocation and Usage Statistics	182
Stream Buffers	183
I/O Buffer Cache	184
RPC Group	185
NFS Group	186
Windows-Specific Groups	188
NT System Group	188
NT Thread Group	190
NT Registry Group	191
NT Service Group	192
NT System Performance Group	193
NT Cache Performance Group	194
NT Memory Performance Group	195
NT Page File Performance Group	197
NT Event Monitor Group	197
NT Registry and Performance Extension Group	199
Unsupported MIB Objects on Windows	199
Monitor Table	202
Process Monitor Table	203
Process Group Monitor Table	203
Log Monitor Table	204
History Table	205
History Sampling Examples	206
Disk Statistics Group	208
Enabling Collection of Disk-Performance Statistics	209
CPU Statistics Group	211
Extension Group	212

Chapter 8 Private Enterprise Traps 215

Format of Trap PDUs	215
monitor Trap	216
monitorEntryNotReady Trap	217
logMonMatch Trap	218
logMonNotReady Trap	219
ntEventMonMatch Trap	220
ntEventMonNotReady Trap	221
monitorClear Trap	222
processStop Trap	223
processStart Trap	224
processThreshold Trap	225
processClear Trap	226
license Trap	227
addrChangeTrap	228
procGroupChangeTrap	229
SNMP Trap Format	230

Chapter 9 Host Resources MIB 233

Host Resources System Group	234
Host Resources Storage Group	236
Host Resources Device Group	236
Device Table	237
Processor Table	238
Disk Storage Table	239
Partition Table	239
File System Table	240
Host Resources Running Software Group	241
Host Resources Installed Software Group	242
Unsupported MIB Objects on Windows Systems	243

Chapter 10	Configuring Threshold Monitoring	245
	Threshold Monitoring.....	245
	The Monitor Table.....	246
	Sample Entry in the Monitor Table.....	246
	Columns of the Monitor Table.....	247
	Optimizing Row Creation.....	251
	Monitor Table Flags.....	252
	Monitor Table Actions.....	255
	Viewing the Monitor Table with AdvantEDGE View.....	256
	Assigning Entry Rows for the Monitor Table.....	257
	Setting Local Policy.....	257
	Reserving Blocks of Rows.....	258
	Configuring the Monitor Table.....	258
	Initial Configuration During Startup.....	259
	Dynamic Configuration During Operation.....	261
	Adding Entries with the monitor Directive.....	261
	Threshold Monitoring Examples.....	264
	Monitoring the 1-Minute Load Average.....	264
	Monitoring the 5-Minute Load Average.....	265
	Monitoring the 15-Minute Load Average.....	266
	Monitoring the System's Interrupt Rate.....	267
	Monitoring the System's Page-Fault Rate.....	268
	Monitoring Number of Incoming Packets on the Interface.....	270
	Monitoring Number of Outgoing Packets on the Interface.....	270
	Monitoring Number of SNMP Packets Received.....	271
	Monitoring Space on the Root File System.....	271
	Monitoring Space on the /usr File System.....	272
	Monitoring the Number of Processes.....	274
	Using the edgemon Utility to Monitor Thresholds.....	274
	edgemon Commands for Threshold Monitoring.....	276
	edgemon Examples.....	278
	Removing Threshold Monitoring Entries.....	279
	Removing Entries from the sysedge.cf File.....	279
	Removing Entries with the edgemon Utility.....	279
	Removing Entries Manually.....	280

Chapter 11 **Configuring Process and Service Monitoring 281**

Monitoring Processes and Windows Services.....	281
Monitoring Windows Services	281
Sample Process Monitor Table Entry.....	282
The Process Monitor Table	283
Columns of the Process Monitor Table	284
Process Attributes.....	287
Optimizing Row Creation	289
Process Monitor Table Flags.....	289
Process Monitor Table Actions	296
Viewing the Process Monitor Table with	
AdvantEDGE View	298
Assigning Entry Rows for the Process Monitor Table	298
Configuring the Process Monitor Table	298
Dynamic Configuration During Operation.....	299
Initial Configuration During Startup	300
Monitoring a Process to Make Sure It Is Running	302
Using the watch process Directive to Monitor	
Process Attributes.....	304
Process Monitoring Examples.....	306
Using the edgewatch Utility to Monitor Processes.....	310
edgewatch Commands for Process Monitoring	312
edgewatch Examples.....	313
Removing Process Monitoring Entries	315
Removing Entries from the sysedge.cf File.....	315
Removing Entries with the edgemon Utility	316
Removing Entries Manually	316
Recommendations for Process and Service Monitoring	317

Chapter 12 **Configuring Process Group Monitoring 319**

Monitoring Process Groups	319
The Process Group Monitor Table	320
Columns of the Process Group Monitor Table	320
Optimizing Row Creation	324
Process Group Monitor Table Flags	325
Process Group Monitor Table Actions	326

Viewing the Process Group Monitor Table with AdvantEDGE View	327
Assigning Entry Rows for the Process Group Monitor Table ...	327
Configuring the Process Group Monitor Table	328
Dynamic Configuration During Operation	328
Initial Configuration During Startup	329
Monitoring a Process Group.	329
Process Group Monitoring Examples	330
Removing Process Group Monitoring Entries	331
Removing Entries from the sysedge.cf File	331
Removing Entries with the edgemon Utility	332
Removing Entries Manually.	332

Chapter 13 Configuring Log File Monitoring 333

Monitoring Log Files	333
Log Monitor Table.	334
Columns of the Log Monitor Table	335
Optimizing Row Creation	337
Log Monitor Table Flags	337
Log Monitor Table Actions.	340
Viewing the Log Monitor Table with AdvantEDGE View.	342
Configuring the Log Monitor Table.	343
Initial Configuration During Start-Up	343
watch logfile Examples	344
Dynamic Configuration During Operation	345
Using the edgewatch Utility to Monitor Log Files	345
edgewatch Commands for Log File Monitoring	346
Sample Uses of the edgewatch Utility	348
Removing Log Monitoring Entries.	349
Removing Entries from the sysedge.cf File	349
Removing Entries with the edgemon Utility	350
Removing Entries Manually.	350
Recommendations for Log File Monitoring	351
Monitoring Log File Size	354
Rotating Log Files	355

Chapter 14	Configuring Windows Event Monitoring	357
	Monitoring Windows Events	357
	Monitoring Windows Event Logs	358
	Checking Log File Status	358
	Search Criteria	359
	NT Event Monitor Table	360
	Columns of the NT Event Monitor Table	360
	Optimizing Row Creation	363
	NT Event Monitor Table Flags	363
	NT Event Monitor Table Actions	366
	Viewing the NT Event Monitor Table with	
	AdvantEDGE View	368
	Configuring the NT Event Monitor Table	368
	Dynamic Configuration During Operation	369
	Initial Configuration During Start-Up	369
	watch ntevent Directive Examples	371
	Using the edgewatch Utility to Monitor Windows Events	372
	edgewatch Commands for NT Event Monitoring	373
	Sample Uses of edgewatch for Monitoring Windows Events	375
	Removing NT Event Monitoring Entries	376
	Removing Entries from the sysedge.cf File	377
	Removing Entries with the edgemon Utility	377
	Removing Entries Manually	377
 Chapter 15	 Configuring History Collection	 379
	History Collection	379
	History Sampling	380
	History Control Table and the Data Table	380
	Columns of the History Control Table	381
	Columns of the History Table	382
	Optimizing Row Creation	383
	Viewing the History Control Table with AdvantEDGE View	384
	Configuring the History Control Table	384
	Initial Configuration During Start-Up	385
	Dynamic Configuration During Operation	386
	Using the emphistory Utility	387
	emphistory Utility Examples	389

Chapter 16	Adding Custom MIB Objects	391
	Systems Management MIB Extension Group	391
	Features of the Extension Group	392
	Configuring Extension Variables	393
	Using the extension Keyword	393
	Additional Parameters	394
	Extension Examples.....	396
	DNS Domain (UNIX Only).....	396
	NIS Domain Name (UNIX Only).....	396
	Remote Pinger (UNIX and Windows)	397
	Writing Extension Scripts	397
	Using Extension Variables with Your Management Software...	399
	Editing empire.asn1 for Extension Variables	400
	Editing a Separate MIB Specification for Extension Variables.....	400
	Recommendations for Using Extensions.....	401
 Chapter 17	 Adding Windows Registry and Performance MIB Objects	 403
	Systems Management MIB ntRegPerf Group	403
	Windows Registry and Performance Functionality	404
	Registry Data	405
	Performance Data	405
	Configuring Windows Registry and Performance Variables...	408
	Using the ntRegPerf Keyword	408
	Windows Registry and Performance Examples.....	409
	CrashControl DumpFile.....	409
	Objects Threads.....	410
	TCP Segments Sent/Sec	410
	Using Windows Registry and Performance Variables with Your Management Software	410
	Editing empire.asn1 for ntRegPerf Variables	411
	Editing a Separate MIB Specification for ntRegPerf Variables.....	411

Chapter 18	Deploying the SystemEDGE Agent	413
	Introduction.....	413
	Deploying SystemEDGE with AdvantEDGE View.....	414
	How the Automated Deployment Works	414
	Deploying SystemEDGE from the Web	415
	Deploying SystemEDGE through E-mail	416
	Third-Party Deployment Tools	416
	Automating Deployment	417
	Making Software Available to Remote Systems.....	417
	Installing Software on Remote Systems	419
	Configuring Software for Distributed Systems	419
	Licensing Software Remotely.....	420
	Security Issues.....	420
 Chapter 19	 Troubleshooting and Usage Suggestions	 423
	Using diagsysedge.exe	423
	Determining Whether the Agent Is Running.....	423
	Obtaining a Report for Troubleshooting	424
	Common Problems and Questions	425
	Agent Not Responding to SNMP Requests	425
	Management System Not Receiving SNMP Trap Messages ..	428
	SystemEDGE Agent Not Accepting Valid License.....	429
	Agent Does Not Run on A Particular Operating	
	System Version	430
	Bind Failed: Address Already In Use	433
	Updating the Monitor Configuration File	433
	Automatically Restarting Processes	436
	Implementing Trap Severity Levels	436
	Required and Recommended System Patches.....	437

Appendix A	Error Messages	439
	SystemEDGE Agent Error Messages	439
	action execution failed	439
	authenFailure src: A.B.C.D community X	439
	bad ioconfig magic	440
	bad pid X for write_runStatus	440
	bad shellOutput directory X.	440
	bad shellOutput file X.	440
	bind failed	440
	block size for X is 0, using 1K	441
	cant open kmem	441
	cannot find title index for.	441
	cannot locate disk pstat data, diskStatsTable not supported	441
	cant open kmem	441
	cant open socket for mib-2	442
	caught SIGHUP.	442
	config file error in subprogram_user_name directive.	442
	config syntax error, line X	442
	Could not find a valid license for machine X.	442
	couldn't fork sub-shell	442
	counterType X not supported. Entry will not be added.	443
	CreateEvent for traps failed	443
	createMutex failed for query mutex	443
	createMutex failed for result mutex	443
	/dev/lan is missing. SystemEDGE cannot continue.	444
	dkiotime read failed, no disk stats	444
	discovering HPUX devices by hand	444
	empire_agent_init failed.	444
	error parsing X in monitor file, line Y.	444
	event regcomp failed, X *desc_filt*	444
	executing subprograms as group X	445
	executing subprograms as user Y.	445
	execv failed for action.	445
	execv failed for extension command.	445
	extension command file X is not a regular file.	445
	extension filename too long.	446
	extension variable X already in use	446
	failed to add monitor entry index X	446

failed to alloc anIDE struct	446
failed to alloc space for monitor	446
failed to create a trap session	447
failed to create timer event, X	447
failed to create a trap session	447
failed to create timer event	447
failed to create trap pdu	447
failed to get dkscinfo	448
failed to get domain name	448
failed to get service handle	448
failed to open /dev/netman	448
failed to open /system	448
failed to open /var/adm/sw/products	449
failed to open /var/sadm/patch	449
failed to open /var/sadm/pkg	449
failed to open config file X	450
failed to open ioconfig	450
failed to open ip for mib2	450
failed to open kmem	450
failed to open mnttab file	450
failed to open/create mon file	451
failed to open openprom device	451
failed to parse config file	451
failed to push ARP for mib2	451
failed to push TCP for mib2	452
failed to push UDP for mib2	452
failed to read ioconfig magic	452
failed to read monitor file	452
failed to reload utmp cache	452
failed to rename mon file	453
failed to allocate history entry	453
failed to send COLDSTART trap	453
fork failed for extension command	453
fork failed for monitor action	453
fork failed for logmonitor action	454
FPE signal caught	454
ID is X,Y	454
identical threads IDs	454

invalid extension variable access mode	454
invalid extension variable type X	455
invalid history description	455
invalid history object type	455
Invalid monitor table index	455
invalid monprocess regular expression	456
invalid NT event log name	456
invalid NT event type	456
invalid number history buckets	456
invalid SNMP variable type X	456
lock of mnttab lock failed	456
license file /etc/sysedge.lic not found	457
license file not found	457
log file is not regular	457
log filename too long	457
logmon entry X re-initialized	457
logmon regcomp failed, X	457
logmon trap entry not ready Index:X	458
logmon trap Index:X	458
logmonitor action execution failed	458
malloc of trap contents failed	458
monitor action execution failed	458
monitor entry X not ready	459
monitor trap Index:X	459
monprocess requires regular expression	459
nlist of /unix failed	459
no extension variable found for X	460
non-existent object to track history of	460
no process matching expression	460
not querying serial port status	460
not sending authen failure traps	461
not stat'ing disks devices	461
not stat'ng floppy devices	461
not stating NFS filesystems	461
not supporting actions	461
not supporting remoteShell group	461
not supporting user/group tables	462
not supporting who table	462

nteventmon entry X not ready	462
odm_initialize failed.	462
openProcess failed on pid	462
openProcessToken failed on pid	463
openprom device not supported.	463
perfDiskObjects X != Num_Disks	463
realloc of mnt cache failed!	463
recvfrom failed	463
reload_process_table: open /proc failed.	463
reload_process_table: proc ioctl failed	464
root device ptr failed, no openprom.	464
sent SIGKILL to process X.	464
sent signal X to process X.	464
setLogmonEntry: invalid set (logfile), row of status	465
setMonEntry: bad size for OID val	465
setMonEntry: invalid oper.	465
setMonEntry: invalid oper type.	465
setMonEntry: invalid stype	465
setMonEntry: invalid type for OID.	465
setMonEntry: invalid type for val	465
setMonEntry: oid type invalid.	466
setMonEntry: stype type invalid	466
stat logfilename failed.	466
stat of extension command file X failed	466
stat of logmon action X failed	466
stat of monfilesystem action X failed.	466
stat of monprocess action X failed	467
stat of nteventmon action X failed	467
sysedge using port X, config file Y.	467
system call ret error X	467
This agent binary is compiled for X, not Y.	467
timeGetDevCaps failed, exiting.	468
timeKillEvent failed	468
trap ipaddress/hostname X invalid	468
turning off process table support	468
turning off sets to Empire process table	468
two processes with PID X	469
two software packages with same index	469

unable to open monitor file	469
unable to process acl for community X	469
unknown HP CPU type	470
unknown NT event log name	470
unknown NT event type.....	470
unknown service start type.....	470
unknown system type	470
username X not found, all subprograms will be disabled ...	471
Using config file	471
Using monitor file.....	471
using old config file.....	471
using old monitor file X; updates will be placed in Y	471
Command-line Utility Error Messages	472
edgemon Error Messages	472
edgewatch Error Messages	474
emphistory Error Messages	478
nteventmon Error Messages.....	479
sendtrap Error Messages.....	481
snmpget Error Messages.....	483
sysvariable Error Messages.....	484
walktree Error Messages.....	485
xtrapmon Error Messages	486

Appendix B Using the syslog Facility 489

Logging syslog Messages	489
Creating a Log File for Daemon Messages.....	491
Creating a Daemon Log File for Sun SPARC Systems.....	492
Creating a Daemon Log File for HP-UX Systems	492

Appendix C	Adding Self-Monitoring Entries to the sysedge.mon File	493
	SystemEDGE Table Backing Store.....	493
	Adding Monitor Table Entries to the sysedge.mon File	494
	Sample Monitor Table Entries in sysedge.mon	497
	Monitoring 1-Minute Load Average.....	497
	Monitoring File Systems	498
	Adding Process Monitor Table Entries to the sysedge.mon File.	499
	Sample Process Monitor Entries in sysedge.mon.....	501
	Monitoring the Netscape Process Run Status	502
	Monitoring the Netscape Process Size	502
	Adding Process Group Monitor Table Entries to the sysedge.mon File	503
	Sample Process Group Monitor Entry in sysedge.mon	505
	Monitoring the httpd Process Group.....	505
	Adding Log Monitor Table Entries to the sysedge.mon File	506
	Sample Log Monitor Entry in sysedge.mon	507
	Monitoring for Failed su Attempts.....	507
	Adding NT Event Monitor Table Entries to the sysedge.mon File	508
	Sample NT Event Monitor Entries in sysedge.mon	510
	Monitoring for Application Errors.....	510
	Adding History Control Table Entries to the sysedge.mon File .	511
	Sample History Control Table Entries in sysedge.mon	512
	Disk Transfer History.....	512
Appendix D	Textual Conventions for Row Status	513
	RFC 1443: Textual Conventions for SNMPv2	513
	Conceptual Row Creation	516
	Interaction 1: Selecting an Instance-Identifier.....	516
	Interaction 2: Creating the Conceptual Row	517
	Interaction 3: Initializing Non-defaulted Objects	520
	Interaction 4: Making the Conceptual Row Available	521
	Conceptual Row Suspension	522
	Conceptual Row Deletion.....	522

Appendix E	Bibliography	525
	Index	529

Preface

This guide supports SystemEDGE Release 4.2 Patchlevel 4 and later. It describes how to install and use eHealth™ SystemEDGE™ on the following platforms:

- Sun™ Solaris™ SPARC Releases 2.6 through 2.10
- Sun Solaris x86 Releases 2.6-2.10 (32-bit)
- Sun Solaris 2.10 (AMD64)
- Microsoft® Windows NT® 4.0 (Service Pack 6a or later), Windows® 2000, Windows XP (x86 only), and Windows 2003 (x86/AMD64/EMT64/Itanium®)
- HP-UX™/PA-RISC 11.0, 11i, and 11i v2 (11.23)
- HP-UX™/Itanium® 11i v2 (11.23)
- AIX™ Releases 4.3.3 and 5.1 through 5.3 (32- and 64-bit)
- Digital® UNIX 4.0F and 4.0G, and Tru64™ 5.1A and 5.1B
- Red Hat™ Linux™ Releases 6.0 through 9.0; Red Hat Enterprise Linux AS 2.1 (x86); Red Hat Enterprise Linux WS/ES/AS 3.0 (Itanium®/x86), Red Hat Enterprise Linux WS/ES/AS 4.0 (Itanium/AMD64/EMT64/Itanium)
- SuSE Linux 9 (x86/AMD64/EMT64/Itanium)
- SUSE Linux Enterprise Server 10 (x86/AMD64/EMT64/Itanium)

When instructions for the operating systems differ, this guide specifies to which operating system the instructions apply.

NOTE

Throughout this guide, the terms Windows and Windows NT encompass Windows NT 4.0, Windows 2000, Windows XP, and Windows 2003. If you are using Windows NT 4.0, you must be using SP 6a or later.

Audience

This guide is intended for an administrator who is installing, configuring, and using the SystemEDGE agent to manage UNIX and Windows workstations. It assumes that you have a basic familiarity with your system's operating system environment and with the Simple Network Management Protocol (SNMP).

About This Guide

This section describes the changes and enhancements that have been made since the last release of this guide. It also includes the documentation conventions used in this guide.

Revision Information

This section describes the changes in this guide since SystemEDGE 4.2 Patchlevel 3.

Added information about new operating system support to the Preface and Chapter 2:

- Sun Solaris 2.10 (AMD64).
- HP-UX/PA-RISC 11.0, 11i, and 11i v2 (11.23).
- SUSE Linux Enterprise Server 10 (x86/AMD64/EMT64/Itanium).

Documentation Conventions

Table 1 lists the conventions used in this document.

Table 1. Documentation Conventions

Convention	Description
File or Directory Name	Text that refers to file or directory names.
code	Text that refers to system, code, or operating system command line examples.
<i>emphasis</i>	Text that refers to guide titles or text that is emphasized.
enter	Text that you must type exactly as shown.
Name	Text that refers to menus, fields in dialog boxes, or keyboard keys.
New Term	Text that refers to a new term, that is, one that is being introduced.
<i>Variable</i>	Text that refers to variable values that you substitute.
→	A sequence of menus or menu options. For example, File → Exit means “Choose Exit from the File menu.”
NOTE	Important information, tips, or other noteworthy details.
CAUTION	Information that helps you avoid data corruption or system failures.
WARNING	Information that helps you avoid personal physical danger.

Technical Support

If you have a Support Contract ID and password, you can access our Support Express knowledgebase at the following URL: <http://search.support.concord.com>.

If you have a software maintenance contract, you can obtain assistance with *eHealth*. For online technical assistance and a complete list of primary service hours and telephone numbers, contact Technical Support at <http://support.concord.com>.

Current SystemEDGE Product Information

For the most current information about the SystemEDGE agent and related products, refer to the SystemEDGE page on the *eHealth* product Web site at <http://www.concord.com/sysedge>.

Introduction

eHealth® SystemEDGE™ increases the productivity of system administration staff by enabling them to control all workstations on their networks from a single, central location. The agent extends management beyond the network boundary and into attached systems to automate systems-management tasks and inventory tracking—functionality that is necessary for increasing productivity and system stability—while helping to reduce rising system-support costs. You can use the SystemEDGE agent to distribute management tasks to the host systems.

Introducing *eHealth* SystemEDGE

The SystemEDGE agent provides powerful system management through the industry-standard SNMP. It enables remote management systems to access important information about the system's configuration, status, performance, users, processes, file systems and much more. In addition, the agent includes intelligent self-monitoring capabilities that enable reporting and managing of exceptions and that eliminate the need for excessive polling.

Microsoft® Data Center Certification

The SystemEDGE agent does not touch the system kernel, whether it runs on a Unix™, Linux™, or Windows® system. So SystemEDGE does not require Data Center Certification.

Using eHealth SystemEDGE

To use the SystemEDGE agent, you must first install it on every workstation or server that you want to monitor. You can then configure it to monitor that system for variables that you specify. The SystemEDGE agent interoperates with SNMP network management system (NMS) platforms, such as eHealth, Aprisma SPECTRUM™, Sun™ Domain Manager™, HP® OpenView, IBM™ NetView 6000™, Micromuse® Netcool™, and others. In addition, the SystemEDGE agent supports the ability to monitor objects from several management information bases (MIBs).

Supported MIBs

A MIB is a virtual information store in which an agent stores information about the elements under its control. Each item of management information is represented by an object, and the MIB is a structured collection of these objects.

A management system monitors a managed resource by reading the values of its MIB objects. It can also control the resource by modifying (setting) the values of objects in the resource's MIB through SNMP commands.

MIBs are defined in a MIB specification that describes the management objects relating to a particular resource. The MIB specification also defines how the collection of objects is structured. The MIB module resembles a data-definition document that is used by both the management system and the agent.

The SystemEDGE agent supports the following MIBs:

- MIB-II (RFC 1213)
- Host Resources MIB (RFC 1514)
- Systems Management MIB

MIB II

MIB-II is the standard that provides information about network interfaces and protocol statistics. This MIB includes information about the following protocols:

- Internet Protocol (IP)
- Transfer Control Protocol (TCP)

- Internet Control Message Protocol (ICMP)
- User Datagram Protocol (UDP)
- Simple Network Management Protocol (SNMP)

For more information about MIB-II, refer to RFC 1213.

Host Resources MIB

The Host Resources MIB is defined by the Internet Engineering Task Force (IETF) to provide management information for generic host systems. The MIB includes information that is especially useful for asset management, such as the following:

- Storage areas, such as file systems and disk partitions
- Running and installed software
- System devices, such as keyboards, disks, and network cards

For more information about how the SystemEDGE agent uses the Host Resources MIB, refer to Chapter 9, “Host Resources MIB.”

Systems Management MIB

The Systems Management MIB is a private-enterprise MIB that includes objects for monitoring the health and performance of the underlying system and its applications. This MIB defines management information for the following:

- Kernel and system parameters
- Boot configuration
- Network, streams, and I/O buffer statistics
- Network file system (NFS) and Remote Procedure Call (RPC) statistics
- Kernel performance statistics, such as the number of context switches and page faults
- File systems
- Mounted devices
- Users

- Processes
- Interprocess communications
- System resources

Table 2 describes the self-monitoring tables that are provided in the Systems Management MIB. These tables configure the SystemEDGE agent's autonomous self-monitoring and data-storage capabilities.

Table 2. Systems Management MIB Monitor Tables

Table	Description
Monitor table	Specifies MIB objects that the SystemEDGE agent monitors and compares to user-specified thresholds.
Process Monitor table	Specifies processes that the SystemEDGE agent monitors for status (whether they are running) and resource utilization.
Process Group Monitor table	Specifies groups of processes that the SystemEDGE agent monitors for status and resource utilization.
Log Monitor table	Specifies regular expression strings for which the SystemEDGE agent searches through user-specified log files.
NT Event Monitor table	Specifies event logs that the SystemEDGE agent searches for specific events.
History Control table	Specifies the sample interval and number of samples for MIB objects that the SystemEDGE agent monitors and stores in the History table for future retrieval by the management system.

For more information about how the SystemEDGE agent uses the Systems Management MIB, refer to Chapter 7, “Systems Management MIB.”

SystemEDGE Self-Monitoring Features

When you manage a large enterprise network with hundreds of systems, you may need to place limits on the information that is monitored, the poll rate, and even the number of systems that are managed. The unique self-monitoring capability of the SystemEDGE agent is specifically designed to provide the kind of management by exception that is necessary in distributed network environments.

The SystemEDGE agent provides the following types of monitoring:

- Threshold monitoring
- Process and service monitoring
- Process group monitoring
- Log file monitoring
- Windows event monitoring
- History collection

Threshold Monitoring

The SystemEDGE agent can monitor exception conditions automatically, reducing or eliminating the need for constant polling by an network management system (NMS). You can configure the agent’s flexible Monitor table to monitor any integer-based MIB object that the agent supports. You set the polling interval, comparison operator (greater than, equal to, and so on), and threshold value, and SystemEDGE automatically monitors the MIB objects that you specify. You can tailor entries for time over threshold to reduce noise. SystemEDGE can also send traps to an NMS if exceptions occur.

For example, you can configure SystemEDGE to monitor the available space on a particular file system and to notify the NMS when the file system becomes too full. For more information, refer to Chapter 10, “Configuring Threshold Monitoring.”

Process and Service Monitoring

With SystemEDGE, you can monitor process attributes for mission-critical processes, Windows services, and applications. For example, you can monitor whether a process is running, the network I/O, system calls, and other attributes.

If any processes stop running, SystemEDGE can automatically notify the NMS and restart them, if necessary. You can configure SystemEDGE to monitor processes in the Process Monitor table. On Windows systems, SystemEDGE can also monitor Windows services. For more information, refer to Chapter 11, “Configuring Process and Service Monitoring.”

Process Group Monitoring

You can use the Process Group Monitor table to define a set of processes that the SystemEDGE agent can track. SystemEDGE can track the number of processes (by name and arguments) that match the regular expression you specified. It can also indicate when a process group changes through the processGroupChange trap. In addition, it can match processes by user name and group name. For more information, refer to Chapter 12, “Configuring Process Group Monitoring.”

Log File Monitoring

SystemEDGE can monitor any ASCII-based system or application log file for regular expressions. For example, you can configure the SystemEDGE agent to monitor the log file `/var/adm/sulog` for a regular expression that you specify. Whenever a message that matches the regular expression you specified is logged to the file, SystemEDGE notifies the NMS through an SNMP trap and includes the log entry that matched the expression. For more information, refer to Chapter 13, “Configuring Log File Monitoring.”

Windows Event Monitoring

SystemEDGE can also monitor Windows event logs for important event types, event identifiers, or events that match specific regular expressions. Whenever an event that matches the search criteria occurs, SystemEDGE notifies the NMS through an SNMP trap. For more information, refer to Chapter 14, “Configuring Windows Event Monitoring.”

History Collection

You can configure the SystemEDGE History Control Table to sample MIB variables and to use the collected data for baselining and trend analysis of your system without having to constantly poll from the NMS. SystemEDGE collects the data, and the NMS can periodically retrieve the history. For more information about configuring SystemEDGE history collection, refer to Chapter 15, “Configuring History Collection.”

NOTE

SystemEDGE history collection capability is short-term only. Use eHealth for long-term history collection. For more information about using eHealth to monitor your systems, refer to the *eHealth — System and Application Administration Guide*.

Identifying Top Processes

SystemEDGE provides a flexible architecture that supports the addition of plug-in modules for monitoring processes and applications. One of these plug-ins is the Top Processes application insight module (AIM), through which the agent can report on processes that are consuming the most CPU resources at any given time. You can use Top Processes to detect and isolate the CPU-dominating processes. Then, you can reallocate resources to ensure optimal system and

application availability and performance. The Top Processes AIM ships with the SystemEDGE agent and does not require a separate license. You can enable this AIM during the SystemEDGE agent installation.

Tracking Assets

You can use SystemEDGE to automate asset tracking and provide an up-to-date picture of your installed hardware and software. SystemEDGE can determine whether your systems are properly configured and whether they include the current patches and service packs. This information can help simplify system management, improve performance, and reduce security risks. For more information about tracking assets, refer to “Inventory Tracking and Asset Management” on page 242.

Supporting Custom MIB Objects

The SystemEDGE agent enables you to create your own scalar MIB objects for customized management. You can configure SystemEDGE with each MIB object’s number and type and the name of a script or program to execute when the new MIB variable is queried or set. For more information, refer to Chapter 16, “Adding Custom MIB Objects.”

Supporting Windows Registry and Perfmon Extensions

You can also configure the SystemEDGE agent for Windows to report additional registry parameters and performance data without using external programs or scripts. This feature allows you to monitor the health of applications that make performance data available through the performance registry. For more information, refer to Chapter 17, “Adding Windows Registry and Performance MIB Objects.”

Specifying Corrective Actions

The SystemEDGE agent can automatically respond to critical situations on a system by invoking **actions**, which are specific commands or scripts that the agent can run automatically when configured to do so. For example, you can specify actions that enable the agent to restart a failed process, send e-mail to an administrator in the event of an unauthorized access to the system, and so on. You can also configure the SystemEDGE agent to perform corrective actions in response to traps. For example, you can configure the agent to run a script or program when a variable's value crosses a specified threshold, or when the agent discovers specific matches on regular expressions in log files or Windows event logs.

When using actions, you must specify the full path of the command (along with any parameters). The agent executes this command each time the conditions are met for the monitoring table entry. If you do not specify an action, the agent does not call a command or script in response to meeting the conditions you specified. For sample actions, refer to the `/contrib` subdirectory of the SystemEDGE agent installation and to the SystemEDGE contributed information Web page on the eHealth Support Web site (support.concord.com).

NOTE

Do not use Windows batch files for actions; they impose severe programmatic limitations and often do not work correctly with desktop applications. Instead, use a more powerful and flexible scripting language, such as Perl or Visual Basic.®

For more information about specifying actions, refer to the following sections:

- “Monitor Table Actions” on page 255
- “Process Monitor Table Actions” on page 296
- “Process Group Monitor Table Actions” on page 326
- “Log Monitor Table Actions” on page 340
- “NT Event Monitor Table Actions” on page 366

SystemEDGE in Windows Clustered Environment

Clusters are groups of servers and other resources that function as a single system to enable high availability and shared workload. Clusters can protect against failure of applications, services, and hardware (including CPUs and disk drives).

The SystemEDGE agent can operate in a Windows cluster to monitor individual components on the physical servers in Windows clusters based on MIB objects in the Systems Management MIB (empire.asn1).

SystemEDGE's NT Event Monitoring can be used to send a trap upon a cluster failover event.

The Systems Management MIB provides basic information about the cluster with the following MIB objects:

- ntIsClustered
- ntClusterName
- ntClusterMembers
- ntClusterIsActive
- ntClusterActiveNode

For more information about these objects, refer to the Systems Management MIB (empire.asn1) in the /doc subdirectory of your SystemEDGE agent distribution.

You can view cluster data by running an AdvantEDGE View™ System Information query. You can also use AdvantEDGE View to apply a custom SystemEDGE template for monitoring clusters to your systems. For more information, see the AdvantEDGE View Web Help.

NOTE

While SystemEDGE can monitor the health of individual servers, monitoring clustered environments requires additional intelligence to distinguish between a failure and a failover. SystemEDGE cannot be used in these environments.

To monitor clustered environments, use CA Unicenter NSM with its Advanced Systems Management option.

Using eHealth AdvantEDGE View

If you are using the SystemEDGE agent with the eHealth AdvantEDGE View element manager, you can run queries on the data collected by SystemEDGE agents through the AdvantEDGE View Web-based graphical user interface. AdvantEDGE View can also automate deployment, licensing, and configuration of your SystemEDGE agents.

You can access AdvantEDGE View from the Systems & Apps tab of the eHealth Web interface. Figure 1 shows the AdvantEDGE View interface.

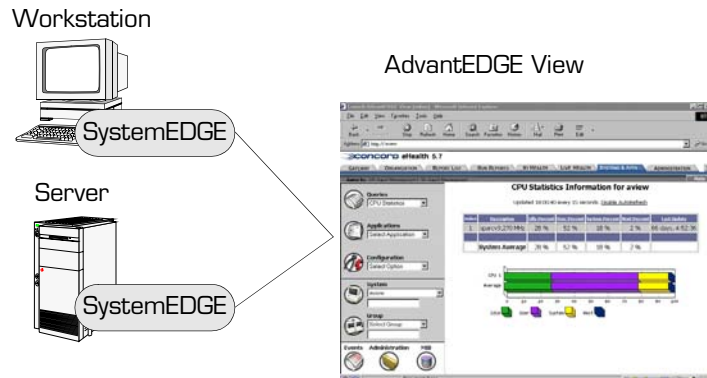


Figure 1. AdvantEDGE View

Using eHealth Service Availability

eHealth Service Availability is a plug-in to the SystemEDGE agent that provides management and monitoring of the response time and availability of the following Internet services:

- Active Directory
- Dynamic Host Configuration Protocol (DHCP)
- Domain Name System (DNS)
- File Input/Output (I/O)
- File Transfer Protocol (FTP)

- Hypertext Transfer Protocol (HTTP) and secure HTTP (HTTPS)
- Internet Message Access Protocol (IMAP)
- Lightweight Directory Access Protocol (LDAP)
- Messaging Application Program Interface (MAPI – Windows only)
- Network Information System (NIS/NIS+)
- Network News Transfer Protocol (NNTP)
- Packet internetwork groper (PING)
- Post Office Protocol (POP3)
- Round-Trip E-mail
- Simple Mail Transfer Protocol (SMTP)
- Simple Network Management Protocol (SNMP)
- SQL Query
- TCP Connect
- Trivial File Transfer Protocol (TFTP)

You can also test custom scripts with the Custom test and WinTask record-playback scripts with the Virtual User test. For more information, refer to the Service Availability Web Help.

Monitoring Voice and Call Quality

eHealth for Cisco CallManager (CCM) and *eHealth* Voice Quality Monitor (VQM) are plug-ins to the SystemEDGE agent that you can use with *eHealth* and AdvantEDGE View to monitor your Cisco CallManager systems and applications, as well as voice quality and jitter on response paths. For more information, refer to the Web Help for CCM and VQM.

Using the eHealth Application Insight Modules

The SystemEDGE agent provides a plug-in architecture through which it can load optional eHealth AIMs when it initializes. These eHealth AIMs (previously called AdvantEDGE Point modules) provide an extensible and flexible approach to supporting application-specific semantic knowledge. Table 3 lists the existing AIMs and the applications for which they provide management.

Table 3. eHealth AIMs

Module	Manages and Monitors
eHealth AIM for Microsoft® Exchange	Microsoft Exchange application
eHealth AIM for Microsoft IIS	Microsoft IIS application
eHealth AIM for Microsoft SQL Server™	Microsoft SQL Server application
eHealth AIM for Apache	Apache Web server
eHealth AIM for Oracle®	Oracle database and application
eHealth AIM for Check Point™ FireWall-1®	Check Point FireWall-1 application
eHealth AIM for Network Services for UNIX	Vital network services for UNIX systems, including Sendmail, DNS, Lightweight Directory Access Protocol (LDAP), NFS, Network Information Services (NIS), and Dynamic Host Configuration Protocol (DHCP)
eHealth AIM for Network Services for Windows	Vital network services for Windows systems, including Active Directory, DHCP, DNS and Windows Internet Naming Service (WINS).

For more information about these AIMs, refer to the user guide for the module in which you are interested.

Using eHealth with eHealth SystemEDGE

You can use the SystemEDGE agent to monitor your eHealth systems, and you can use the agent and eHealth together to manage and monitor other systems within your enterprise.

When you are using the SystemEDGE agent to monitor eHealth, you can run the `nhAddSysEdgeMonEntries` command to configure the agent to monitor critical eHealth processes and system logs. The command adds entries to the `sysedge.cf` file, and it stops and restarts the SystemEDGE agent to implement the changes. For more information about the `nhAddSysEdgeMonEntries` command, refer to the *eHealth Administration Reference*.

When you use SystemEDGE with eHealth, the eHealth poller can collect data from SystemEDGE agents and store that data in the eHealth database. The information is then available to the eHealth reporting and real-time monitoring tools. You can run At-a-Glance (AAG), Trend, Top N, What-If Capacity Trend, System Health, and MyHealth reports for systems to perform capacity planning, accurately document service problems, and troubleshoot potential problems. For more information, refer to the *eHealth — System and Application Administration Guide*. Figure 2 shows a sample AAG report for systems.

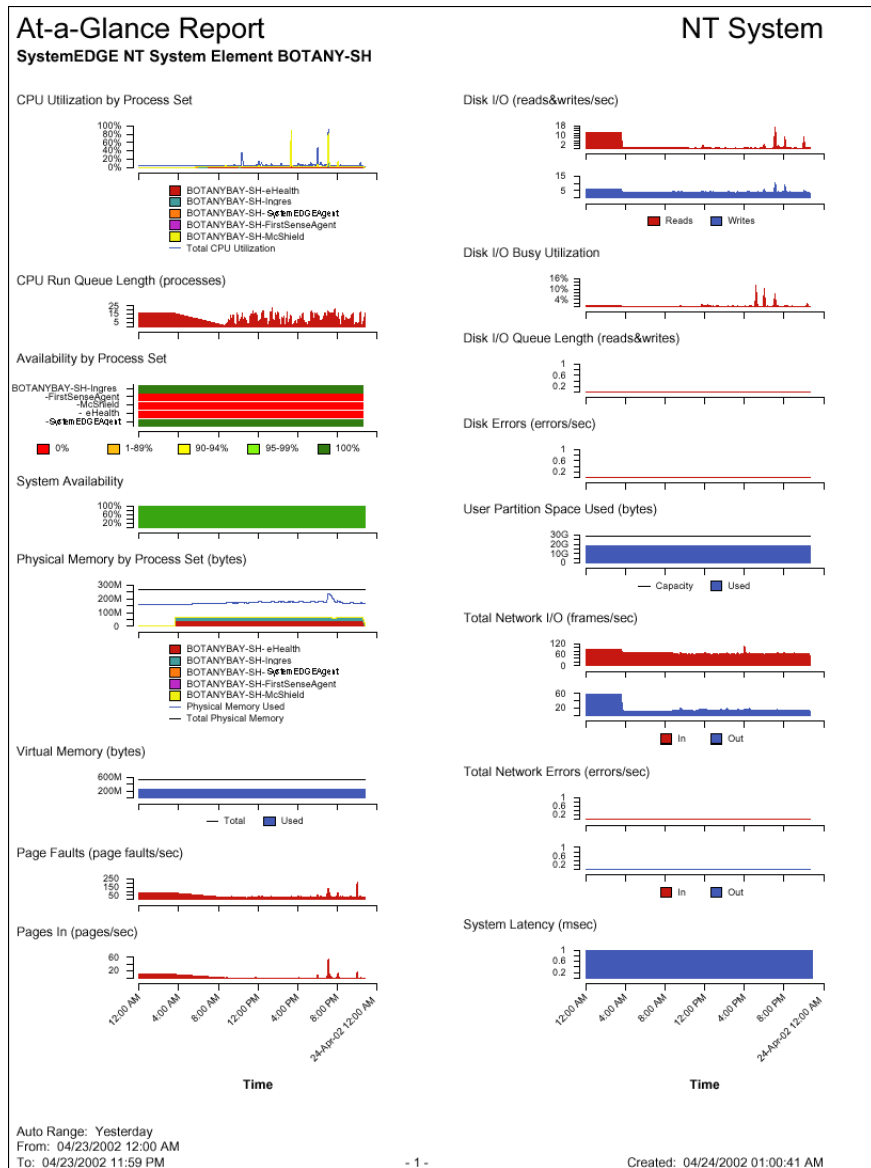


Figure 2. Sample AAG Report for Systems

Using eHealth Live Health — Fault Manager with eHealth SystemEDGE

You can also use the SystemEDGE agent with Live Health™ — Fault Manager for real-time detection of potential problems. Live Health extends the features of eHealth to provide real-time performance and availability management for applications, systems, and networks. The SystemEDGE agent collects performance data, and Live Health analyzes the data with unique algorithms to identify outages and delays.

When you are using the SystemEDGE agent with Live Health, you can use the default Live Exceptions profiles for the SystemEDGE agent and the eHealth AIMs, or you can define your own profiles. The profiles organize alarm variables by delay, availability, unusual workload, and latency.

Fault Manager is an enhancement to Live Exceptions that enables eHealth to receive SNMP trap messages from devices and systems. Fault Manager interprets and processes trap information, reduces the noise of duplicate and repeated messages, and alerts you to the problems and conditions that interest you. When the eHealth system receives a trap, it processes the trap based on Live Exceptions rules and profiles that you configure. Thus, you can configure Fault Manager to raise an alarm for the associated element, or to ignore various trap messages.

NOTE

You can edit the `sysedge.cf` file to configure the SystemEDGE agent to feed specific traps to Fault Manager. For more information, refer to “Configuring Trap Communities” on page 130.

Figure 3 shows how Fault Manager collects data from SNMP trap sources, such as the SystemEDGE agent, and sends it to a variety of displays.

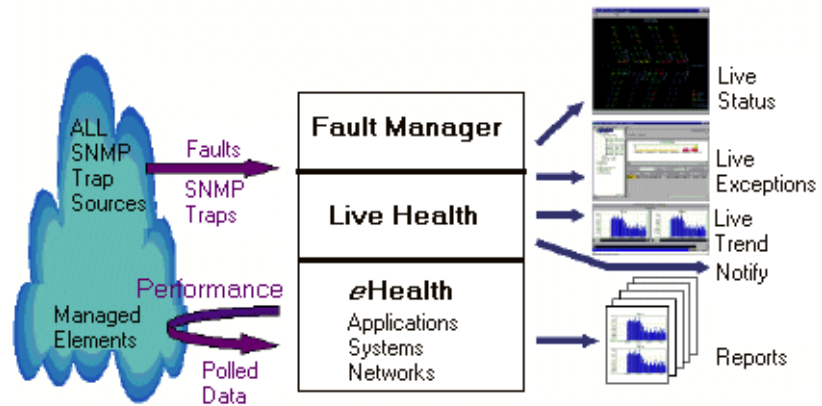


Figure 3. eHealth and Live Health – Fault Manager

Guidelines for Using the SystemEDGE Agent

You can gain the most value from the SystemEDGE agent by setting up effective management policies for your systems, networks, and applications. In particular, follow these guidelines:

- Use the SystemEDGE agent with AdvantEDGE View and eHealth. For more information, refer to the *eHealth AdvantEDGE View User Guide* and the *eHealth Administration Guide*.
- Automate management tasks through scheduling. If you are using SystemEDGE with eHealth, refer to the *eHealth Administration Guide* for more information.

- Limit SNMP access to the agent through access control lists and binding to private interfaces.
 - Use read-write communities for SNMP Sets and read-only communities for querying and polling. For more information about specifying community strings, refer to “Configuring Access Communities” on page 128.
 - Use either port 161 (the SNMP industry standard port) or 1691 (which is reserved with the Internet Assigned Numbers Authority [IANA] for use with the SystemEDGE agent). For more information about configuring ports, refer to “Configuring the SNMP Bind Address” on page 143.
- Create groups in your management software based on the following:
 - Operating systems
 - Database systems
 - Clients
 - File servers
 - E-mail servers
 - Web servers
 - Physical location

For more information about creating groups, refer to the *eHealth Administration Guide*.

- Define policies across groups of systems instead of on an individual system basis. Use management software such as AdvantEDGE View to push rules to groups of systems. For more information, refer to the *eHealth AdvantEDGE View User Guide*.
- Create a standard configuration for each group (through AdvantEDGE View Template Management or manually in the sysedge.cf file) that is based on system roles (for example, one configuration for e-mail servers and another for Web servers; one for UNIX systems and another for

Windows systems, and so on). Apply that configuration through AdvantEDGE View or by copying `sysedge.cf` to each system that you are monitoring. For more information about applying SystemEDGE Configuration templates, refer to the *eHealth AdvantEDGE View User Guide*. For more information about the `sysedge.cf` file, refer to Chapter 4, “Configuring the SystemEDGE Agent.”

- Include meaningful information in the system location and contact field of your `sysedge.cf` file. For example, include information such as Rack 0, Slot 1, Atlanta and e-mail: `it@yourdomain.com`.
- Use a standard table of row indexes across your self-monitoring tables. For example, use rows 10,000 to 10,999 across all self-monitoring tables for threshold monitoring, rows 11,000 to 11,999 for process monitoring, and so on. When defining these indexes and reserving rows, keep in mind the following:
 - Use large ranges of index numbers for each type of monitoring to allow for growth.
 - Use standard index entries for specific types of monitoring entries. For example, always use row 10,000 for monitoring the total amount of CPU available.
 - Use AdvantEDGE View Template Management to apply your self-monitoring entries to groups of systems.

For more information about using standard index rows, refer to “Assigning Entry Rows for the Monitor Table” on page 257. For more information about AdvantEDGE View, refer to the *eHealth AdvantEDGE View User Guide*.

- Base any de-duplication on source index number or matched string within the trap—not on trap type alone.

- Keep the following points in mind as you create entries that you can use across multiple systems:
 - Monitor total CPU utilization states (which allow the configuration to be portable across single- and multi-processor systems).
 - Monitor thresholds and configure SystemEDGE to send a limited number of traps (for example, two or three) to prevent flooding of the NMS.
 - Enable Clear events to specify resetting of the event in the agent and the status in an NMS.

For more information about effectively monitoring thresholds and clearing traps, refer to “Monitor Table Flags” on page 252.

- When you are managing a large network (hundreds or thousands of systems) and polling each system for granular data (at intervals of less than 15 minutes), do the following:
 - Use history collection to gain highly-granular data at the agent level, and let the management system poll the History table. You can use SystemEDGE for short-term history collection and eHealth (or another management system) for the long-term historical view. For more information, refer to Chapter 15, “Configuring History Collection.”
 - Push the monitoring out to the agent, and configure the agent to send traps based on these self-monitoring entries.
- Limit the number of potential traps from a single monitoring entry by using the following SystemEDGE flags as you set up your self-monitoring entries:
 - 0x00000200: Send traps only after *x* occurrences of this event.
 - 0x00000400: Send up to *x* traps for this entry.

For more information about traps, refer to “Monitor Table Flags” on page 252.

For more usage tips, refer to the following sections in this guide:

- “Recommendations for Configuring Security” on page 146
- “Recommendations for Process and Service Monitoring” on page 317
- “Recommendations for Log File Monitoring” on page 351
- “Recommendations for Using Extensions” on page 401

In addition, refer to the SystemEDGE Contributions page at <http://www.concord.com/sysedge-contrib/>.

Simple Network Management Protocol

SNMP is a standard for managing TCP/IP-based networks and devices. A typical network management environment contains many **managed devices**—each with an agent process—and at least one NMS, also referred to as the Manager or management system. The management system sends messages to the agent processes on the managed devices to request information or to modify parameters. The agent process carries out the management system’s request and returns a reply. Additionally, the agent can send its own messages (traps) to the management system to notify it of important events. SNMP is the protocol that the agent and management system use to exchange this management information.

SNMP Message Types

The SystemEDGE agent uses standard SNMP messages to exchange management information with management systems. Table 4 describes the SNMP message types.

Table 4. SNMP Message Types

Message Type	Description
GetRequest	An NMS sends a GetRequest to an agent to obtain the value of a specific object-instance from the MIB.
GetNextRequest	An NMS sends a GetNextRequest to an agent to obtain the next object instance.
GetResponse	An agent returns the requested information to an NMS through a GetResponse.
SetRequest	An NMS sends a SetRequest to instruct an agent to change the value of a MIB object's parameters.
Trap	An agent sends a trap to notify an NMS of exceptions.

SNMP Communities

In SNMP, a **community** is a relationship between an agent and any number of management systems; it defines authentication and access-control permissions for communication between the management systems and the agent. An SNMP community is identified by a string of octets called the **community name**, which appears in the header portion of every SNMP message.

The agent checks the community name in the SNMP message header to determine if the message is authentic. If the community name matches one that is accepted by the agent, the message is considered to be authentic, and the agent processes the message. If it does not match, the agent records an

authentication failure and drops the message. The community name also serves to determine what level of access (read-only or read-write) is available when the agent is using that community name.

NOTE

Although the community name is similar to a password, providing access to read—and even change—the values of an agent’s MIB objects, the community name is *not* encrypted when it appears in an SNMP message header; it appears in clear text.

IP spoofing occurs when a system impersonates a trusted system to gain access to another system.

Although it is possible to attach IP-based, access-control lists to individual communities, IP spoofing can circumvent the access control lists. Consequently, you should consider SNMP Version 1 (SNMPv1) communities insecure and take configuration steps to limit potential security violations. In addition, you should improve the overall security of the distributed system through router and system configuration. For more information, refer to Chapter 4, “Configuring the SystemEDGE Agent.”

Access Communities

Upon installation, the SystemEDGE agent is configured with only one community, named **public**. This community provides read-only access to the agent’s MIB object values. For security reasons, the default configuration does not define a read-write community. (The agent’s configuration file contains a sample read-write community, but it is commented out.) Before you can modify (set) the agent’s MIB values, you must define a community that provides read-write access. For information about configuring your own SNMP communities for the agent, refer to “Configuring Access Communities” on page 128.

Trap Communities

The **Trap community** indicates which management systems should receive the trap messages that the agent generates. The agent's self-monitoring features report exception conditions by sending trap messages to the management systems in the trap community. For information about configuring the SystemEDGE agent to send traps, refer to “Configuring Trap Communities” on page 130.

For information about the types of traps that the SystemEDGE agent can send, refer to Chapter 8, “Private Enterprise Traps.”

Installing the SystemEDGE Agent

This chapter explains how to install the SystemEDGE agent. Use Table 5 to find the installation procedure for your operating system.

Table 5. Installation Instructions By Operating System

Operating System	Section of this Chapter
Solaris	"Installing SystemEDGE on Solaris Systems" on page 52
Windows	"Installing SystemEDGE on Windows Systems" on page 59
HP-UX	"Installing SystemEDGE on HP-UX Systems" on page 75
Linux	"Installing SystemEDGE on Linux Systems" on page 80
AIX	"Installing SystemEDGE on AIX Systems" on page 82
Digital UNIX and Tru64	"Installing SystemEDGE on Digital UNIX and Tru64 Systems" on page 84

Installing SystemEDGE on Solaris Systems

This section describes how to install the SystemEDGE agent on Solaris systems. See the Preface for a list of supported systems and versions.

These instructions explain how to install SystemEDGE in the `/opt/EMPsysedge` directory. If you want to install SystemEDGE in a different directory, refer to “Installing SystemEDGE in a Non-Default Directory for Solaris” on page 58.

System Requirements

Before you begin installing the agent, verify that your system meets the following requirements:

- CD-ROM drive
- At least 8 MB RAM
- At least 4 MB free disk space

Adding the SystemEDGE Agent Package

The SystemEDGE agent for Solaris 2.x is distributed as a software **package**. This distribution uses the standard `pkgadd` utility to install the agent.

To install the software:

1. Log in as root by entering **su** and then the root password at the password prompt.
2. Insert the CD that contains the software distributions into the CD-ROM drive. If you are running volume management, the CD is mounted automatically. If you are not running volume management, you must mount the CD by entering the following command:

```
mount -r -t hsfs /dev/rdisk/c0t6d0s2 /cdrom
```

For more information about volume management, refer to the man page for `vold(1M)`.

NOTE

The CD-ROM device depends on whether the device is Integrated Drive Electronics (IDE) or Small Computer System Interface (SCSI), and on the particular SCSI ID setting (if applicable). For more information about the CD-ROM device, refer to your system documentation.

3. Run the `pkgadd` utility to install the software.

For Solaris SPARC systems, enter the following:

```
pkgadd -d /cdrom/sysedge/sol
```

For Solaris x86 systems, enter the following:

```
pkgadd -d /cdrom/sysedge/solx86
```

The installation script displays the following message:

```
Concord SystemEDGE Agent post-installation starting
```

When you run the installation script, you must enter valid values for all prompts. If you enter invalid values or press Enter at a prompt that requires a value (and does not offer a default value), the script will not complete properly. If the script generates an error message or is unable to complete, you must run the script again and specify valid values for all prompts.

This script enables you to configure the SystemEDGE agent. You can change the configuration information after the installation if desired; for more information, refer to Chapter 4, “Configuring the SystemEDGE Agent.” The installation script also enables automatic licensing of the agent. For information about additional ways to license the agent, refer to Chapter 3, “Licensing the SystemEDGE Agent.”

NOTE

For each prompt, the default value is shown in brackets ([]). You can press **Return** to accept the default, or you can enter another value and then press **Return**.

4. Disable the native SNMP agent, if desired, by pressing **Return** at the following prompt:

```
Disable the native SNMP Agent (if applicable)? [yes]
```

For more information about running multiple SNMP agents simultaneously, refer to Chapter 6, “Using the SystemEDGE Agent with Other SNMP Agents’.”

5. Configure the SystemEDGE agent to use UDP port 1691 if you are running another SNMP agent on port 161 by entering **y** at the following prompt:

```
Change SystemEDGE port to 1691 (default is 161)?  
[yes]
```

NOTE

You must perform this step if you did not disable the native SNMP agent in the previous step.

6. Press **Return** at the following prompt if you want to enter a description for your system:

```
Configure system description? [yes]
```

- a. Enter the system description, for example, **Test System 1**, at the following prompt:

```
Enter system description (followed by newline):
```

7. Press **Return** at the following prompt if you want to enter a location for your system:

```
Configure system location? [yes]
```

- a. Enter the system location, for example, **QA Lab**, at the following prompt:

```
Enter system location (followed by newline):
```

8. Press **Return** at the following prompt if you want to enter the name of a contact for this system:

```
Configure system contact? [yes]
```

- a. Enter the contact name, for example, **Test system contact**, and then press **Return** at the following prompt:

Enter system contact (followed by newline):

9. Press **Return** at the following prompt to configure SystemEDGE to load the Top Processes AIM:

Enable Top Processes AIM? [yes]

The installation script displays the following:

```
Configuring community strings. . . . .
You should configure a read-only and a read-write community.
You need a read-only community to discover SystemEDGE.
You need a read-write community to use auto-magic license
utilities like licenseeme, AdvantEDGE View, etc.
```

10. Press **Return** at the following prompt to configure a read-only community:

Configure a read-only community-string? [yes]

- a. Enter the name of the community that you want to configure as read-only, at the following prompt:

Enter read-only community (no spaces, case-sensitive) [public]:

The script displays the following:

Setting read-only community to public

11. Press **Return** at the following prompt to configure a read-write community:

Configure a read-write community-string? [yes]

- a. Enter the name of the community that you want to configure as read-write (for example, **private**), at the following prompt:

Enter read-write community (no spaces, case-sensitive):

The script displays the following:

```
Setting read-write community to private
```

12. Press **Return** at the following prompt to configure a trap destination:

```
Configure a Trap Destination? [yes]
```

- a. Enter the trap destination IP address (for example, **aview.empire.com**) at the following prompt:

```
Enter trap destination IP address:
```

- b. Enter the trap community, or press **Return** to configure public as a trap community at the following prompt:

```
Enter Trap community [public]:
```

- c. Enter the trap community (for example, **public**) at the following prompt:

```
Enter trap community [public]:
```

The script displays the following:

```
Adding trap community to config file  
Restarting SystemEDGE
```

13. Press **Return** at the following prompt to use the licenseme utility to automatically license the SystemEDGE agent:

```
License SystemEDGE via licenseme utility? [yes]
```

The script displays the following:

```
You will need to have access to http://license.concord.com  
either directly or through a web-proxy; if the proxy requires  
authentication, you will need username/password for it.  
You also need SNMP read-write access to SystemEDGE.  
You also need your licensing account username/passwd.
```


- a. Enter **sysedge** at the following prompt:

```
Product (choose one) [sysedge|svcrsp|xchgmod|iismod|apachemod|oramod|sqlmod]:
```

- b. Enter the type of license you are requesting at the following prompt:

```
License Type (choose one) [eval|permanent|usage-based|partner]:
```

- c. Enter the IP address of the Web proxy server (if applicable) at the following prompt:

```
Web (HTTP) proxy server [none]:
```

- d. Enter the proxy user name (if applicable) at the following prompt:

```
Web (HTTP) proxy username [none]:
```

- e. Enter the password for that proxy user (if applicable) at the following prompt:

```
Web (HTTP) proxy password [none]:
```

- f. Enter the user name for the Web licensing account at the following prompt:

```
Web Licensing Account Username:
```

- g. Enter the password for the Web licensing account at the following prompt:

```
Web Licensing Account Password:
```

- h. Enter your name at the following prompt:

```
Your Name:
```

The installation script displays a message similar to the following:

```
licenseme: licensing for host '127.0.0.1' was successful
Concord SystemEDGE Agent Installation complete.
```

Installing SystemEDGE in a Non-Default Directory for Solaris

By default, the SystemEDGE agent for Solaris is installed in the `opt/EMPsysedge` directory. You can install SystemEDGE in a different directory under the `/opt` directory, but you cannot change the `EMPsysedge` directory name. For example, you can install SystemEDGE in `/opt/Concord/EMPsysedge`.

To install SystemEDGE in a different directory for Solaris:

1. Create an administration text file that specifies the new installation directory, as follows:

```
basedir=/opt/Concord
mail=
instance=unique
partial=ask
runlevel=ask
idepend=ask
rdepend=ask
space=ask
setuid=ask
conflict=ask
action=ask
```

2. Enter the following from a command prompt to instruct the `pkgadd` utility to use the text file you created (*myadmin.file* in this example) to install the agent to the directory you specified:

```
pkgadd -a myadmin.file -d ./sysedge.pkg
```

This example installs the agent in `/opt/Concord/EMPsysedge`.

3. Follow the installation prompts, as described in “Adding the SystemEDGE Agent Package” on page 52.

4. If you are installing any *eHealth* AIMs or other SystemEDGE modules, install them in the *plugins* subdirectory of the directory you specified (for example, `/opt/Concord/EMPsysedge/plugins`).

Installing SystemEDGE on Windows Systems

This section describes how to install the SystemEDGE agent on Windows systems. See the Preface for a list of supported systems and versions.

System Requirements

Before you begin installing the agent, verify that your system meets the following requirements:

NOTE

If you are using Windows NT 4.0, you must be using Service Pack 6a or later. If you are using an earlier service pack, you will receive an error message when you attempt to start the SystemEDGE agent.

- CD-ROM drive
- At least 8 MB memory
- At least 4 MB free disk space

NOTE

Do *not* install SystemEDGE into a directory that includes spaces in the pathname (such as `C:\Program Files`). If you do, the agent will have difficulty locating and loading the *eHealth* AIMs.

Installing the Software with InstallShield

Throughout this guide, the terms Windows and Windows NT encompass Windows NT 4.0, Windows 2000, Windows XP, and Windows 2003.

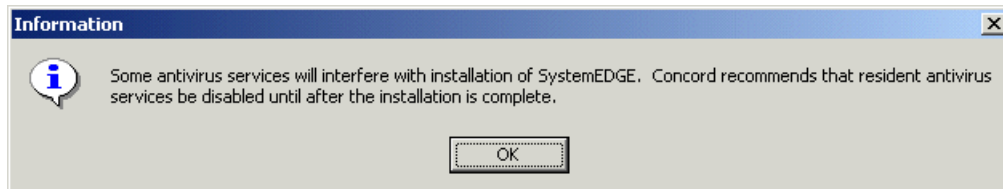
The SystemEDGE agent for Windows is distributed as an InstallShield® program named sysedge.exe.

NOTE

You can also install SystemEDGE from the command line, as described in “Installing the Software from the Command Line” on page 74. If you have previously installed the agent from the command line, you *can* upgrade using the InstallShield program. However, you cannot run the command line installation to upgrade an existing InstallShield installation. If you have installed SystemEDGE with InstallShield and you want to upgrade from the command line, you must uninstall the InstallShield version (as described in “Uninstalling an Agent with InstallShield” on page 99) and then reinstall from the command line.

To install the software:

1. Log on to the Windows system as administrator.
2. Insert the CD that contains the software distributions into the CD-ROM drive. Windows automatically mounts the drive using the CD-ROM drive's corresponding drive letter.
3. Change to the sysedge\win directory, and double-click sysedge.exe to run the InstallShield program. The following dialog box appears to recommend turning off any anti-virus programs currently running.



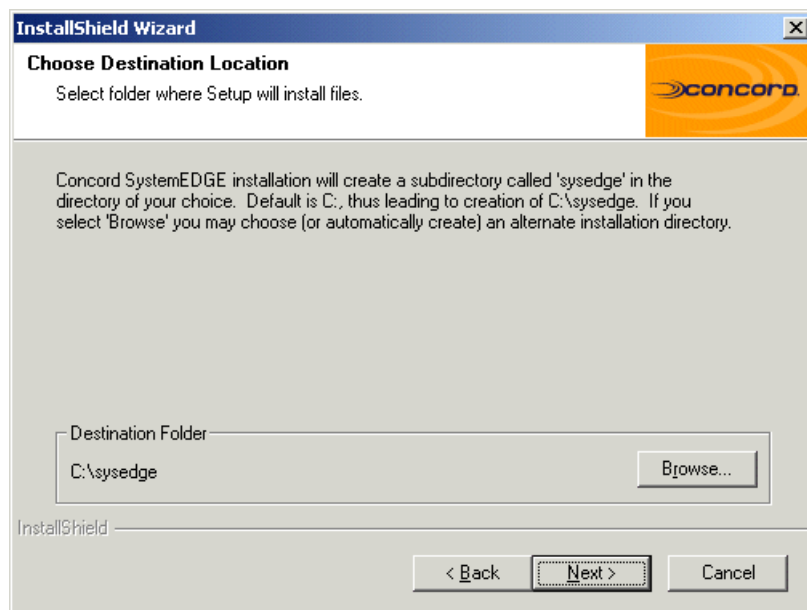
4. After disabling your antivirus program, click OK. The following dialog box appears.



NOTE

If you are upgrading the SystemEDGE agent, the SystemEDGE Maintenance dialog box appears. Select **Modify**, and then click **OK**. After the program updates the agent, click **Finish**.

5. Click **Next**. The Choose Destination Location dialog box appears.

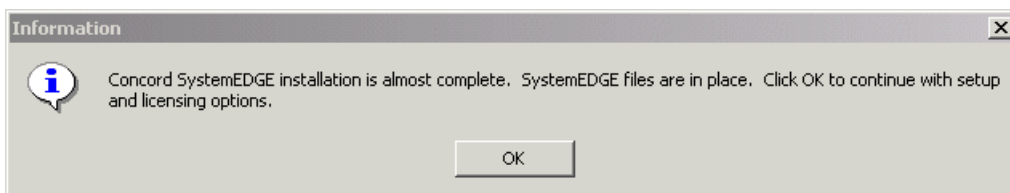


6. Click **Next** to accept the default installation directory, or click **Browse**, select the directory in which to install SystemEDGE, and then click **Next**. The recommended directory is C:\sysedge. You must use the root directory, and you must keep the sysedge directory name, but you can select a different directory. You can, for example, install

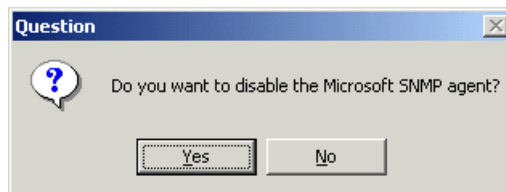
the agent in C:\Concord\sysedge. (If you select a different directory, make sure you install any SystemEDGE plug-ins under the same directory [for example, in C:\Concord\sysedge\plugins].)

The setup program copies the sysedge.dll file and the sysedge.cf, sysedge.mon, and sysedge.lic configuration files into the system root directory, %SystemRoot%\system32. If you are upgrading from a previous version of the SystemEDGE agent, the installation does not overwrite the configuration files. Instead, it copies the new files to the \config subdirectory.

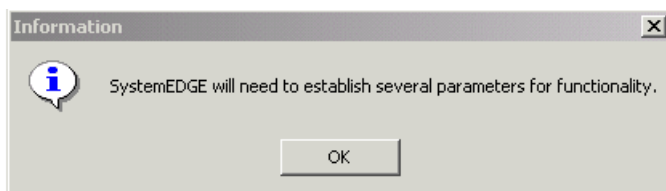
The following dialog box appears.



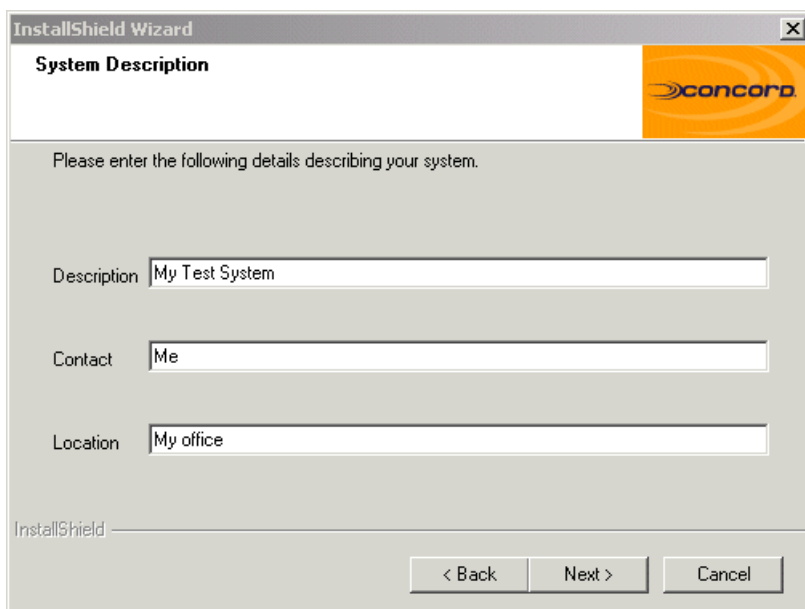
7. Click **OK**. The following dialog box appears.



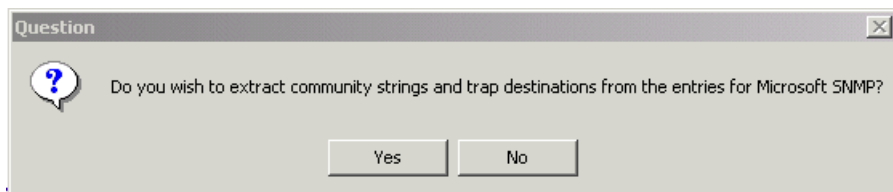
8. Click Yes to disable the Microsoft SNMP agent, or No to run both agents. The following dialog box appears.



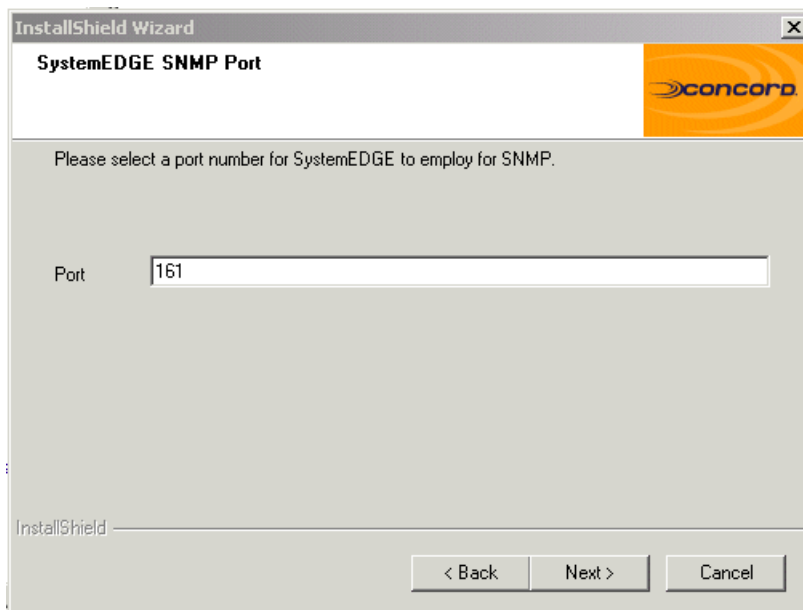
9. Click **OK**. The following dialog box appears.

A screenshot of the "InstallShield Wizard" window, specifically the "System Description" step. The window has a title bar with "InstallShield Wizard" and a close button. Below the title bar is a header area with "System Description" on the left and the "concord" logo on the right. The main area contains the instruction: "Please enter the following details describing your system." Below this are three text input fields: "Description" with the value "My Test System", "Contact" with the value "Me", and "Location" with the value "My office". At the bottom left, it says "InstallShield". At the bottom right are three buttons: "< Back", "Next >", and "Cancel".

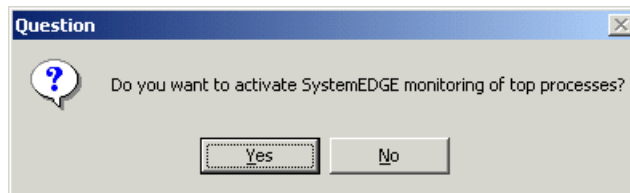
10. Specify a description, contact name, and system location in the **Description**, **Contact**, and **Location** fields, and then click **Next**.



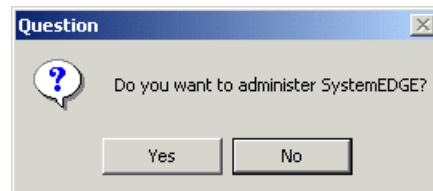
11. Click **Yes** to extract SNMP configuration information from the Microsoft SNMP agent, or **No** if you want to configure the SystemEDGE agent to use different community strings and trap destinations. The following dialog box appears.



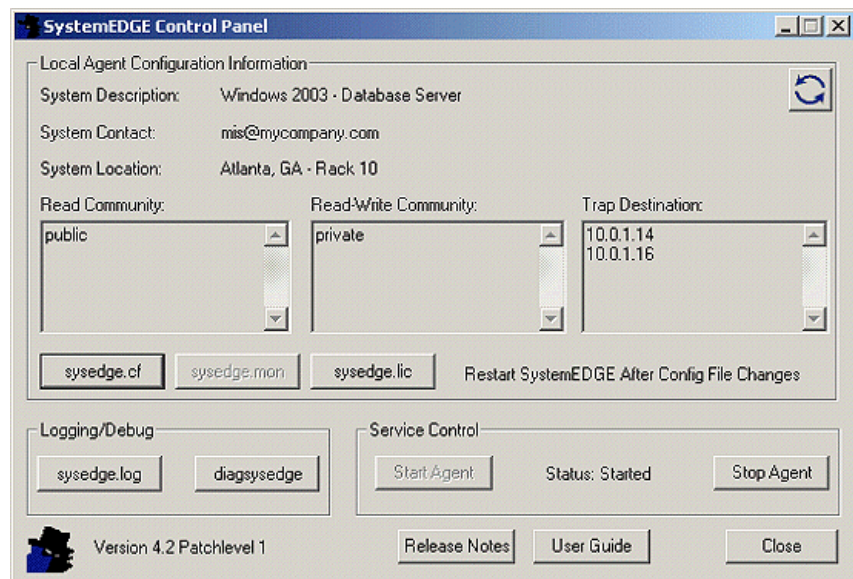
12. Specify the port on which you want the SystemEDGE agent to run in the **Port** field, and then click **Next**. Port 161 is the default SNMP port. If you run SystemEDGE on this port and you are running the Microsoft SNMP agent, you must specify a different port for the Microsoft agent. Port 1691 is reserved for use with the SystemEDGE agent. You can specify any port that is not in use on the system. The following dialog box appears.



13. Click **Yes** to enable the Top Processes AIM. The following dialog box appears.



14. Click **Yes**. The SystemEDGE Control Panel appears.



15. Edit the `sysedge.cf` file:

- a. Click **sysedge.cf**. The file opens in a text editor.
- b. Add community strings or trap destinations. You must add a read-write community string if you want to license the agent automatically. For more information, refer to Chapter 4, “Configuring the SystemEDGE Agent’.”
- c. Save and close the file.

You can also use the Control Panel to do the following:

- Start and stop the agent.
- View community strings and trap destinations in the **Read Community**, **Read-Write Community**, and **Trap Destinations** fields.
- View and edit the configuration and license files by clicking **sysedge.cf**, **sysedge.mon**, or **sysedge.lic**.
- View the SystemEDGE log file by clicking **sysedge.log**.
- Run the **diagsysedge** utility by clicking **diagsysedge**.
- View this guide or the *eHealth SystemEDGE Release Notes* by clicking **Manual** or **Release Notes**.

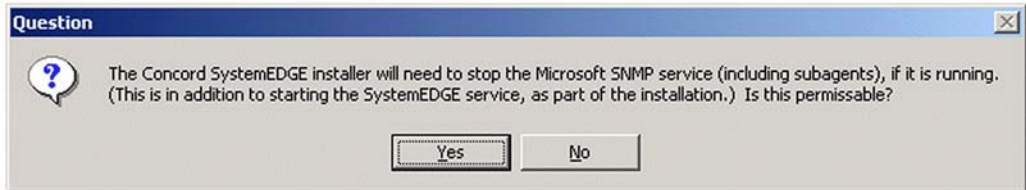
After the installation, you can access the SystemEDGE Control Panel by selecting **Start** → **Settings** → **Control Panel**, and double-clicking **eHealth SystemEDGE**.

16. When you are finished modifying the SystemEDGE configuration, click **Close**.

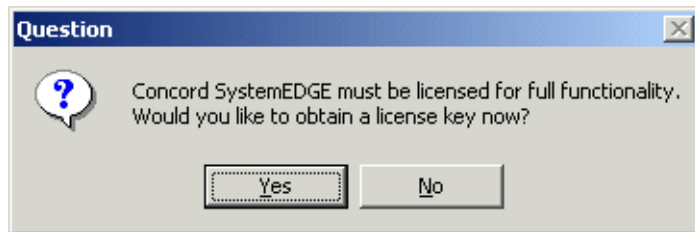
NOTE

If you did not specify a read-write community string, an informational dialog box informs you that you cannot automatically license the agent without specifying a read-write community string. Click **OK**, and then return to the SystemEDGE Control Panel to specify a community string unless you do not want to license the agent automatically.

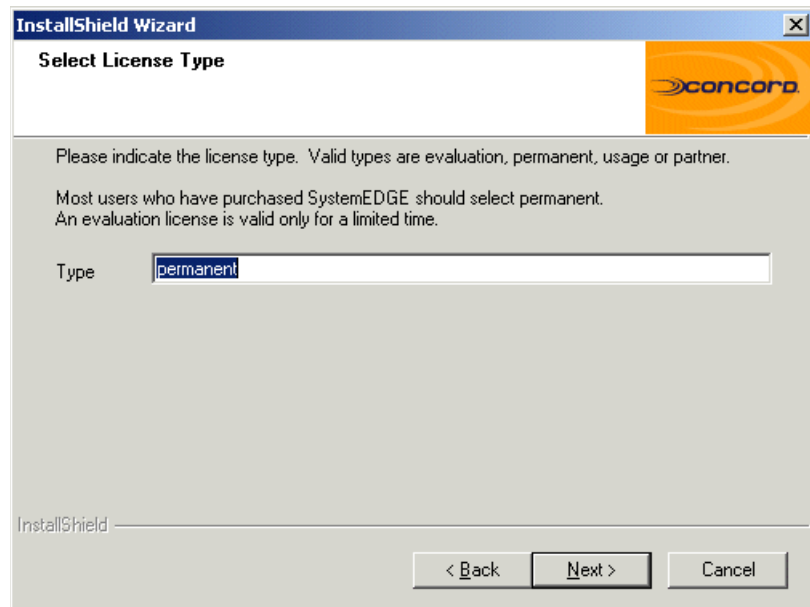
17. The following dialog box appears. You must click **Yes** to continue and complete the installation. If you click **No**, the installation will be aborted.



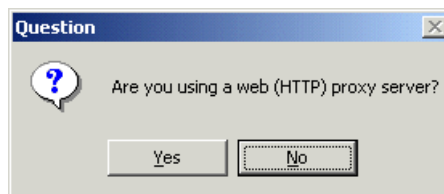
18. After clicking **Yes** in the previous step, the following dialog box appears.



19. Click **Yes** if you want to obtain a license. The Select License Type dialog box appears.



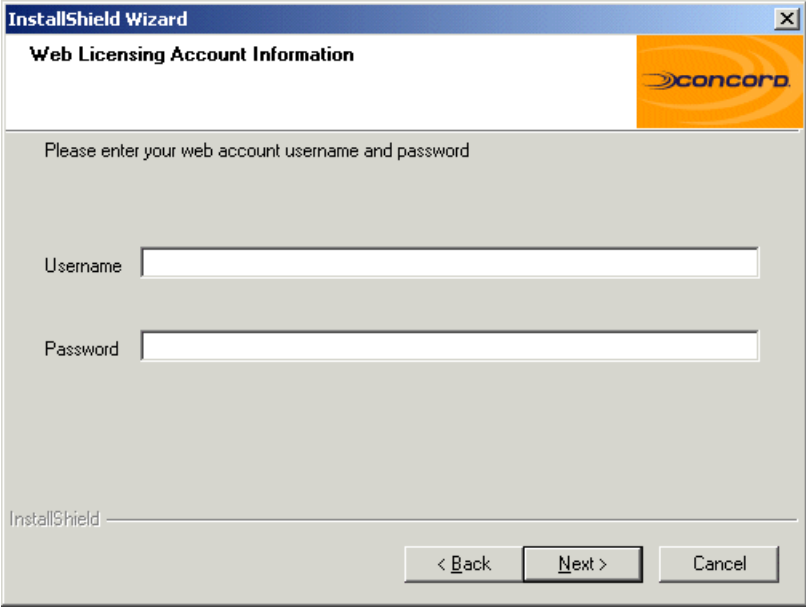
20. Specify the type of license (evaluation, permanent, or usage-based) in the **Type** field, and then click **Next**. The following dialog box appears.



21. Click **Yes** if you are using a Web server. If you do, the Proxy Information dialog box appears.
- Enter the correct information in the **Server**, **Username**, and **Password** fields.

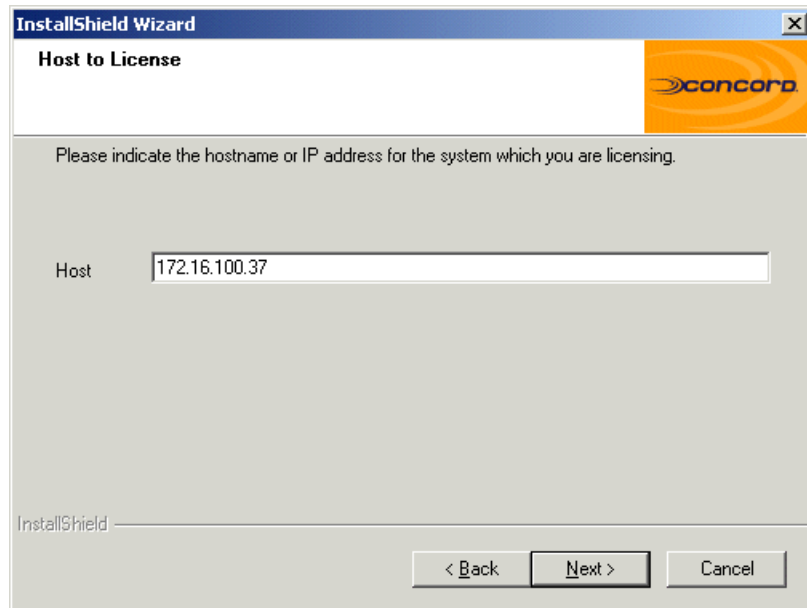
b. Click **Next**.

If you click **No**, the Web Licensing Account Information dialog box appears.



The screenshot shows a Windows-style dialog box titled "InstallShield Wizard". The main heading is "Web Licensing Account Information". In the top right corner, there is an orange square logo with the word "concord" in white. Below the heading, the text "Please enter your web account username and password" is displayed. There are two input fields: "Username" and "Password", each with a text box to its right. At the bottom left, the text "InstallShield" is visible. At the bottom right, there are three buttons: "< Back", "Next >", and "Cancel".

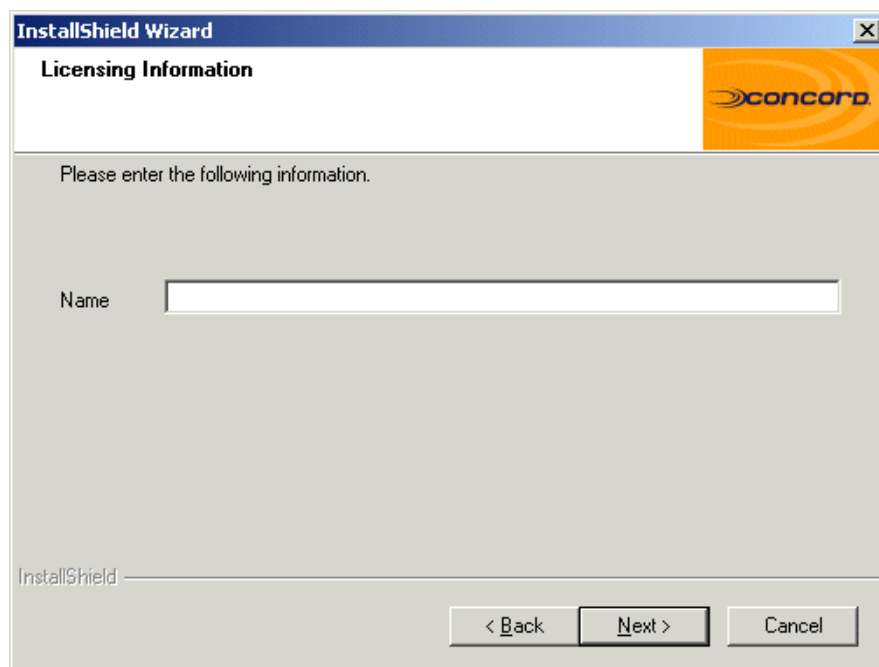
22. Enter your user name and password in the **Username** and **Password** fields, and then click **Next**. The Host to License dialog box appears.



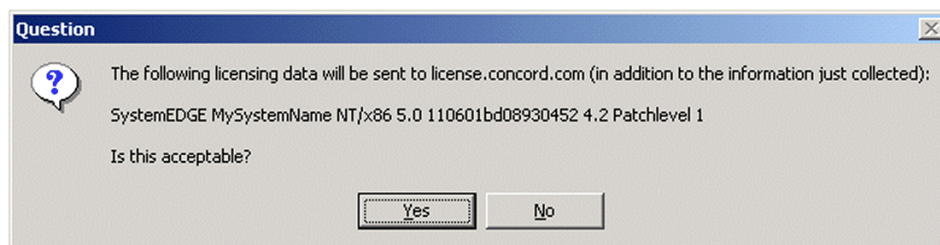
- 23.** Enter the hostname or IP address of the system you want to license in the **Host** Field. The Licensing Information dialog box appears.

NOTE

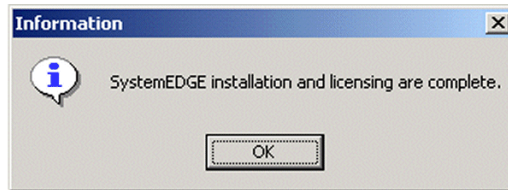
The IP address supplied for licensing must permit SNMP SET operations. If you have an ACL the IP address must match one of the hosts in the ACL or licensing will fail. If a failure occurs, the agent can be licensed separately from the installation process.



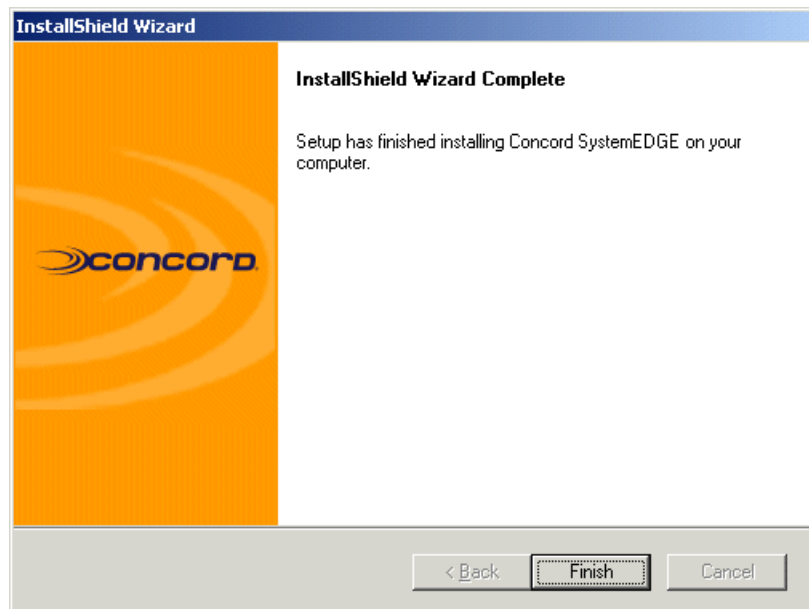
24. Enter your name in the **Name** field, and then click **Next**.
25. After entering the name and clicking next the following dialog box will appear.



26. Click Yes. When licensing completes, the following dialog box appears.



27. Click OK. The following dialog box appears.



28. Click **Finish**.

NOTE

The *only* cases in which the system reboots after you install the agent are the following: if you are running a Windows NT 4.0 operating system with a version of Windows Installer that is earlier than 1.10; or if you are running a

Windows 2000 system that does not include the Windows Installer service. Use the Windows `msiexec` command to determine which version of the Windows Installer is running on your system.

Installing the Software from the Command Line

The SystemEDGE agent command line installation package for Windows is distributed as an executable named `sysedge.exe` in the `wincmd` folder on the AdvantEDGE CD-ROM.

NOTE

You can also install SystemEDGE with InstallShield, as described in “Installing the Software with InstallShield” on page 60. If you have previously installed the agent from the command line, you *can* upgrade using the InstallShield program. However, you cannot run the command line installation to upgrade an existing InstallShield installation. If you have installed SystemEDGE with InstallShield and you want to upgrade from the command line, you must uninstall the InstallShield version (as described in “Uninstalling an Agent with InstallShield” on page 99) and then reinstall using the procedure in this section.

1. Log on to the Windows system as administrator and insert the AdvantEDGE CD-ROM into the CDROM drive. Windows automatically mounts the drive.
2. Run the self-extracting executable by entering the following at the command line (where D: is the CD-ROM drive):

```
D:\sysedge\wincmd\sysedge.exe C:\
```

The agent is installed in C:\sysedge.

Installing SystemEDGE on HP-UX Systems

This section describes how to install the SystemEDGE agent on HP-UX systems. See the Preface for a list of supported systems and versions.

System Requirements

Before you begin installing the agent, verify that your system meets the following requirements:

- CD-ROM drive
- At least 8 MB RAM
- At least 4 MB free disk space

Installing the Software

Hewlett Packard defines a software depot as a group of related file sets.

The SystemEDGE agent software distribution for HP-UX is formatted as an HP **software depot**. This distribution uses the `swinstall` utility to install the agent.

To install the software:

1. Log in as root by entering **su** and then the root password at the password prompt.
2. Insert the CD that contains the software distributions into the CD-ROM drive, and mount it using SAM(1M) as `/cdrom`.
3. Run the `swinstall` utility by entering the following:

```
/usr/sbin/swinstall -s /cdrom/sysedge/hpux/sysedge.dpt EMPsysedge
```

By default, the `swinstall` utility places the software in the `/opt` directory.

4. Run the installation script from the `/opt/EMPsysedge` directory by entering the following commands:

```
cd /opt/EMPsysedge  
./Install
```

When you run the installation script, you must enter valid values for all prompts. If you enter invalid values or press Enter at a prompt that requires a value (and does not offer a default value), the script will not complete properly. If the script generates an error message or is unable to complete, you must run the script again and specify valid values for all prompts.

The installation script displays the following message:

```
Concord SystemEDGE Agent Post-Installation Starting
Determining platform specific information
Copying config, monitor, license files to /etc
Copying startup file to /sbin/init.d and making link
into /sbin/rc2.d
```

You can change the configuration information after the installation if desired; for more information, refer to Chapter 4, “Configuring the SystemEDGE Agent’.” The post-installation script also enables automatic licensing of the agent. For information about additional ways to license the agent, refer to Chapter 3, “Licensing the SystemEDGE Agent’.”

NOTE

For each prompt, the default value is shown in brackets ([]). You can press **Return** to accept the default, or you can enter another value and then press **Return**.

5. Disable the native SNMP agent (if you want to disable it) by entering **yes** at the following prompt:

```
Disable native SNMP agent (if applicable)? [no]
```

For more information about running multiple SNMP agents simultaneously, refer to Chapter 6, “Using the SystemEDGE Agent with Other SNMP Agents’.”

The script displays the following:

```
Disabling native SNMP agent
snmpdm stopped
```

6. Configure SystemEDGE to use UDP port 1691 if you are running another SNMP agent on port 161 by pressing **Return** at the following prompt:

```
Change SystemEDGE port to 1691 (default is 161)? [yes]
```

NOTE

You must perform this step if you did not disable the native SNMP agent in the previous step.

7. Press **Return** at the following prompt if you want to enter a description for your system:

Configure system description? [yes]

- a. Enter the system description, for example, **Test system**, and then press **Return** at the following prompt:

Enter system description (followed by newline):

8. Press **Return** at the following prompt if you want to enter a location for your system:

Configure system location? [yes]

- a. Enter the system location, for example, **Test system location**, and then press **Return** at the following prompt:

Enter system location (followed by newline):

9. Press **Return** at the following prompt if you want to enter the name of a contact for this system:

Configure system contact? [yes]

- a. Enter the contact name, for example, **Test system contact**, and then press **Return** at the following prompt:

Enter system contact (followed by newline):

10. Press **Return** at the following prompt to configure SystemEDGE to load the Top Processes AIM:

Enable Top Processes AIM? [yes]

The installation script displays the following:

```
Enabling topprocs-hpux32bit.so AIM
Configuring community strings:
o You should configure a read-only and a read-write community.
o You need a read-only community to discover SystemEDGE.
o You need a read-write community to use auto-magic license
utilities like licenseeme, AdvantEDGE View, etc.
```

11. Press **Return** at the following prompt to configure a read-only community:

```
Configure a read-only community-string? [yes]
```

- a. Enter the name of the community that you want to configure as read-only at the following prompt:

```
Enter read-only community (no spaces, case-sensitive) [public]:
```

The script displays the following:

```
Setting read-only community to public
```

12. Press **Return** at the following prompt to configure a read-write community:

```
Configure a read-write community-string? [yes]
```

- a. Enter the name of the community that you want to configure as read-write (for example, **private**), at the following prompt:

```
Enter read-write community (no spaces, case-sensitive):
```

The script displays the following:

```
Setting read-write community to private
```

13. Press **Return** at the following prompt to configure a trap destination:

```
Configure a Trap Destination? [yes]
```

- a. Enter the hostname or IP address of the trap destination (for example, `aview.empire.com`) at the following prompt:

Enter trap destination IP address:

- b. Enter the trap community, or press **Return** to configure public as a trap community at the following prompt:

Enter trap community [public]:

The script displays the following:

Adding trap community to config file

Restarting SystemEDGE

14. Press **Return** at the following prompt to use the `licenseme` utility to automatically license the SystemEDGE agent:

License SystemEDGE via licenseme utility? [yes]

The script displays the following:

You will need to have access to `http://license.concord.com` either directly or through a web-proxy; if the proxy requires authentication, you will need username/password for it.
You also need SNMP read-write access to SystemEDGE.
You also need your licensing account username/passwd.

- a. Enter **sysedge** at the following prompt:

Product (choose one) [sysedge|svcrsp|xchgmod|iismod|apachemod|oramod|sqlmod]:

- b. Enter the type of license you are requesting at the following prompt:

License Type (choose one) [eval|permanent|usage-based|partner]:

- c. Enter the IP address of the Web proxy server (if applicable) at the following prompt:

Web (HTTP) proxy server [none]:

- d. Enter the proxy user name (if applicable) at the following prompt:

Web (HTTP) proxy username [none]:

- e. Enter the password for that proxy user (if applicable) at the following prompt:

Web (HTTP) proxy password [none]:

- f. Enter the user name for the Web licensing account at the following prompt:

Web Licensing Account Username:

- g. Enter the password for the Web licensing account at the following prompt:

Web Licensing Account Password:

- h. Enter your name at the following prompt:

Your Name:

The installation script displays the following:

```
licenseme: licensing for host '127.0.0.1' was successful  
Concord SystemEDGE Agent Installation complete.
```

Installing SystemEDGE on Linux Systems

This section describes how to install the SystemEDGE agent on Linux systems. See the Preface for a list of supported systems and versions.

System Requirements

Before you begin installing the agent, verify that your system meets the following requirements:

- Intel or Intel-compatible chipset
- CD-ROM drive
- At least 8 MB RAM
- At least 4 MB free disk space

Installing the Software

The SystemEDGE agent for Linux is distributed as a tar file named `sysedge.tar`.

To install the software:

1. Log in as root by entering **su** and then the root password at the password prompt.
2. Create the home directory for the SystemEDGE agent by entering the following:

```
mkdir /opt/EMPsysedge
```

NOTE

The recommended default installation directory is `/opt/EMPsysedge`. You can use another directory, but the examples throughout this guide assume that you are installing the SystemEDGE agent in the `/opt/EMPsysedge` directory.

This command requires the CD-ROM to be located at `/dev/cdrom`, and the mount directory to be `/cdrom`. For information about the locations of your CD-ROM device and mount point, refer to your Linux documentation.

3. Insert the CD-ROM containing the software distributions into the CD-ROM drive, and mount it with the following command:

```
mount -rt iso9660 /dev/cdrom /cdrom
```

4. Change the directory to the agent's home directory, and load the files from the CD-ROM by entering the following commands, one at a time:

```
cd /opt/EMPsysedge
```

```
tar xvof /cdrom/sysedge/linux/sysedge.tar
```

When you run the installation script, you must enter valid values for all prompts. If you enter invalid values or press Enter at a prompt that requires a value (and does not offer a default value), the script will not complete properly. If the script generates an error message or is unable to complete, you must run the script again and specify valid values for all prompts.

5. Start the installation script by running the installation script from the directory where you installed SystemEDGE. For example, enter the following if you copied the installation files to the /opt/EMPsysedge directory:

```
cd /opt/EMPsysedge
./Install
```

You can change the configuration information after the installation if desired; for more information, refer to Chapter 4, “Configuring the SystemEDGE Agent.” The post-installation script also enables automatic licensing of the agent. For information about additional ways to license the agent, refer to Chapter 3, “Licensing the SystemEDGE Agent.”

NOTE

For a detailed example of the installation script, refer to “Installing SystemEDGE on HP-UX Systems” on page 75. For each prompt, the default value is shown in brackets ([]). You can press **Return** to accept the default, or you can enter another value and then press **Return**.

Installing SystemEDGE on AIX Systems

This section describes how to install the SystemEDGE agent on AIX systems. See the Preface for a list of supported systems and versions.

System Requirements

Before you begin installing the agent, verify that your system meets the following requirements:

- CD-ROM drive
- At least 8 MB RAM
- At least 4 MB free disk space

Installing the Software

The SystemEDGE agent software distribution for AIX is formatted as an AIX product. This distribution uses the `smit` utility to install the SystemEDGE agent product.

To install the software:

1. Log in as root by entering **su** and then the root password at the password prompt.
2. Insert the CD containing the software distributions into the CD-ROM drive, and mount it using the AIX command `smit mountfs`. Mount the CD-ROM as `/cdrom`.

NOTE

You can select another mount point, but if you do, be sure to substitute your mount point in place of `/cdrom` in the following instructions.

By default, the `smit` utility places the software in the `/usr/lpp/EMPsysedge` directory.

When you run the installation script, you must enter valid values for all prompts. If you enter invalid values or press Enter at a prompt that requires a value (and does not offer a default value), the script will not complete properly. If the script generates an error message or is unable to complete, you must run the script again and specify valid values for all prompts.

3. Run the `smit` utility by entering the following:

```
smit install_latest
```

- a. Specify `/cdrom/sysedge/aix` as the software input device.
 - b. Select **EMPsysedge.rte** as the software to install.
 - c. Click **OK**.
4. Start the installation script by entering the following commands:

```
cd usr/lpp/EMPsysedge  
./Install
```

The installation script displays the following message:

```
Concord SystemEDGE Agent post-installation starting
```

This script enables you to configure the SystemEDGE agent. You can change the configuration information after the installation if desired; for more information, refer to Chapter 4, “Configuring the SystemEDGE Agent.” The installation script also enables automatic licensing of the agent. For information about additional ways to license the agent, refer to Chapter 3, “Licensing the SystemEDGE Agent.”

NOTE

For a detailed example of the installation script, refer to “Installing SystemEDGE on HP-UX Systems” on page 75. For each prompt, the default value is shown in brackets ([]). You can press **Return** to accept the default, or you can enter another value and then press **Return**.

Installing SystemEDGE on Digital UNIX and Tru64 Systems

This section describes how to install the SystemEDGE agent on Digital UNIX and Tru64 systems. See the Preface for a list of supported systems and versions

System Requirements

Before you begin installing the agent, verify that your system meets the following requirements:

- CD-ROM drive
- At least 8 MB RAM
- At least 4 MB free disk space

Installing the Software

Digital UNIX and Tru64 defines a product kit as a group of related file subsets.

The SystemEDGE agent software distribution for Digital UNIX and Tru64 is formatted as a **product kit**. This distribution uses the `setld` utility to install the SystemEDGE agent package.

To install the software:

1. Log in as root by entering **su** and then the root password at the password prompt.
2. Insert the CD containing the software distributions into the CD-ROM drive, and mount it using the following command (assuming that your CD-ROM drive is at SCSI address 0):

```
mount -r -t cdfs /dev/rz0c /cdrom
```

By default, the distribution places the software in the /usr/opt/EMPsysedge directory.

3. Run the setld utility by entering the following commands, one at a time:

```
cd /cdrom/sysedge/tru64
cp sysedge.tar /tmp
cd /tmp
tar xvof sysedge.tar
/usr/sbin/setld -l /tmp EMPSEEDGE
```

When you run the installation script, you must enter valid values for all prompts. If you enter invalid values or press Enter at a prompt that requires a value (and does not offer a default value), the script will not complete properly. If the script generates an error message or is unable to complete, you must run the script again and specify valid values for all prompts.

4. Run the installation script by entering the following commands:

```
cd /usr/opt/EMPsysedge
./Install
```

The installation script displays the following message:

```
Concord SystemEDGE Agent post-installation starting
```

This script enables you to configure the SystemEDGE agent. You can change the configuration information after the installation if desired; for more information, refer to Chapter 4, “Configuring the SystemEDGE Agent.” The installation script also enables automatic licensing of the agent. For information about additional ways to license the agent, refer to Chapter 3, “Licensing the SystemEDGE Agent.”

NOTE

For a detailed example of the installation script, refer to “Installing SystemEDGE on HP-UX Systems” on page 75. For each prompt, the default value is shown in brackets ([]). You can press **Return** to accept the default, or you can enter another value and then press **Return**.

Reviewing the Configuration Files

When the SystemEDGE agent first starts, it reads the following configuration text files to determine configuration settings and to check that the system has a valid license key for running the agent:

- sysedge.lic
- sysedge.cf
- sysedge.mon

The SystemEDGE agent installation automatically installs these files in the /etc (UNIX) or %SystemRoot%\system32 (Windows) directories during the installation process, unless you are upgrading from a previous version of the SystemEDGE agent. If you are upgrading, SystemEDGE does *not* overwrite your existing files, but copies the new files to the config subdirectories to enable you to compare the new files with the existing files.

Table 6 lists the chapters in which you can find more information about these configuration files.

Table 6. Configuration Files (Page 1 of 2)

Configuration File	Chapter Containing More Information
sysedge.mon	<ul style="list-style-type: none">• Chapter 10, “Configuring Threshold Monitoring”• Chapter 11, “Configuring Process and Service Monitoring”• Chapter 12, “Configuring Process Group Monitoring”• Chapter 13, “Configuring Log File Monitoring”• Chapter 14, “Configuring Windows Event Monitoring”• Chapter 15, “Configuring History Collection”• Appendix C
sysedge.lic	Chapter 3, “Licensing the SystemEDGE Agent”

Table 6. Configuration Files (Page 2 of 2)

Configuration File	Chapter Containing More Information
sysedge.cf	<ul style="list-style-type: none">• Chapter 4, “Configuring the SystemEDGE Agent”• Chapter 5, “Starting the SystemEDGE Agent”• Chapter 10, “Configuring Threshold Monitoring”• Chapter 11, “Configuring Process and Service Monitoring”• Chapter 12, “Configuring Process Group Monitoring”• Chapter 13, “Configuring Log File Monitoring”• Chapter 14, “Configuring Windows Event Monitoring”• Chapter 15, “Configuring History Collection”• Chapter 16, “Adding Custom MIB Objects”• Chapter 17, “Adding Windows Registry and Performance MIB Objects”

NOTE

CA recommends that you use the `sysedge.cf` file (rather than the `sysedge.mon` file) for manually adding entries to the monitoring tables and configuring the agent. The `sysedge.mon` file is a backing store for the agent’s self-monitoring tables. The two files interact, and entries in `sysedge.cf` file take precedence over entries in `sysedge.mon`. The `sysedge.cf` file is static and cannot be edited remotely. The `sysedge.mon` file is not static, so AdvantEDGE View and other management software can update this file through SNMP Sets.

Configuration Files for UNIX Systems

By default, the SystemEDGE agent looks for the configuration files in the /etc directory. The SystemEDGE agent installation script installs these files into that directory automatically during the installation, unless the directory already includes files with those names (that is, if you are performing an upgrade instead of a new installation). You need to copy these files manually *only* if you installed the SystemEDGE agent as an upgrade. Edit these files to match your local requirements.

NOTE

If you installed the SystemEDGE agent as an upgrade, review the new files in /EMPSysedge/config to identify the latest features, and then integrate them with your current configuration files.

You can also instruct the SystemEDGE agent to look for these files in another directory by specifying an alternate directory through command-line arguments. For more information, refer to Chapter 5, “Starting the SystemEDGE Agent”.

Configuration Files for Windows Systems

By default, the SystemEDGE agent looks for the configuration files in the %SystemRoot%\system32\ directory. The SystemEDGE agent setup program installs these files into that directory automatically during the installation. You do not need to copy these files manually unless you installed the SystemEDGE agent as an upgrade instead of a new installation. Edit these files to match your local requirements.

NOTE

If you installed the SystemEDGE agent as an upgrade, review the new files in \sysedge\config to identify the latest features, and then integrate them with your current configuration files.

SystemEDGE Agent Directories and Files

This section describes the directories and files that the SystemEDGE agent installation creates. Table 7 and Table 8 describe the platform-specific files. Table 9 describes files that are common across platforms.

Directories and Files for UNIX Systems

Table 7 describes the directories and files that the SystemEDGE agent installation creates for UNIX systems only.

Table 7. SystemEDGE Directories and Files for UNIX Systems (Page 1 of 2)

Directory	File	Description
bin/		Subdirectory that contains the SystemEDGE agent daemon, as well as several command-line utilities for use with SystemEDGE threshold, process, process group, log, and Windows event monitoring and history sampling.
	checkfile.exe	Program for checking whether a file exists. You can use this program as an extension or part of an action script with SystemEDGE.
	email.exe	Self-contained program for sending e-mail to phones, pagers, and e-mail systems. You can use this program as an action with SystemEDGE.
	sysedge	SystemEDGE agent executable.
	sendtrap	Command-line utility for sending user-defined traps.
	edgewatch	Command-line utility for dynamically configuring the agent to monitor log files and processes.
	edgemon	Command-line utility for dynamically configuring threshold monitoring.
	emphistory	Command-line utility for use with history sampling; use this utility to specify MIB variables to store over time and to retrieve stored values.
	restartproc.sh	Sample action script that restarts a process.

Table 7. SystemEDGE Directories and Files for UNIX Systems (Page 2 of 2)

Directory	File	Description
bin/ (continued)	sysvariable	Command-line utility for retrieving select MIB variables in conjunction with scripts.
	xtrapmon	Command-line utility for receiving and printing SNMP trap messages.
	walktree	Command-line utility for retrieving select MIB variables.
doc/	Subdirectory that contains documentation for the SystemEDGE agent, the MIB modules it supports, and the command-line utilities that are provided with it.	
	edgemon.1	Man page for the edgemon command-line utility.
	edgewatch.1	Man page for the edgewatch command-line utility.
	emphistory.1	Man page for the emphistory command-line utility.
	sendtrap.1	Man page for the sendtrap command-line utility.
	sysedge.1	Man page for the SystemEDGE agent.
	sysedge.cf.5	Man page that describes how to specify configuration settings for the agent by placing them in the sysedge.cf configuration file.
	sysedge.mon.5	Man page that describes how to specify entries for the agent's self-monitoring tables by placing them in the sysedge.mon configuration file.
	sysvariable.1	Man page that describes how to use the sysvariable utility to query systems and exercise the agent.
	walktree.1	Man page that describes how to use the walktree utility to retrieve specific MIB variables.
	xtrapmon.1	Man page that describes how to use the xtrapmon utility to receive and log SNMP trap messages.

NOTE

You can use the `nroff` command to view Man pages from the command line if your Man path is not set correctly. To do so, enter the following:

```
#nroff -man sysedge.1 | more
```

Directories and Files for Windows Systems

Table 8 describes the directories and files that the SystemEDGE agent installation creates for Windows systems only.

Table 8. SystemEDGE Directories and Files for Windows Systems (Page 1 of 3)

Directory	File	Description
setup.exe	Command line version of the installation program for SystemEDGE. NOTE: You do not need to use this file if you installed SystemEDGE through InstallShield.	
sysedge.cpl	SystemEDGE Control Panel. You can use the SystemEDGE Control Panel to access the configuration and log files, to run <code>diagsysedge.exe</code> , and to start and stop the agent, but you do not have to use it for those tasks. You can also continue to perform those tasks from the command line.	
sysedge.dll	SystemEDGE agent dynamic link library.	
sysedge.exe	SystemEDGE agent executable.	
bin/	Subdirectory that contains the SystemEDGE agent dynamic link library, as well as several command-line utilities for use with SystemEDGE threshold, process, process group, log, and Windows event monitoring and history sampling.	
	sendtrap.exe	Command-line utility for sending user-defined traps.
	bounce.exe	Program for shutting down or rebooting Windows systems. You can use this program as an extension or part of an action script with SystemEDGE.
	checkfile.exe	Program for checking whether a file exists. You can use this program as an extension or part of an action script with SystemEDGE.

Table 8. SystemEDGE Directories and Files for Windows Systems (Page 2 of 3)

Directory	File	Description
bin/ (continued)	edgewatch.exe	Command-line utility for dynamically configuring the agent to monitor Windows event logs, log files, and processes.
	edgemon.exe	Command-line utility for dynamically configuring the agent's threshold monitoring.
	emphistory.exe	Command-line utility for use with SystemEDGE history sampling; you can use this utility to specify MIB variables to store over time and to retrieve stored values.
	email.exe	Self-contained sample e-mail action script for sending e-mail to phones, pagers, and e-mail systems.
	getver.exe	Extension tool for finding version information from .exe or .dll files. Some .exe or .dll files do not provide this information, but most do.
	restartsvc.exe	Action program for restarting Windows services in conjunction with process monitoring.
	sysvariable.exe	Command-line utility for retrieving select MIB variables in conjunction with scripts.
	walktree.exe	Command-line utility for retrieving select MIB variables.
	xtrapmon.exe	Command-line utility for receiving and printing SNMP trap messages.
doc/	Subdirectory that contains documentation for the SystemEDGE agent, the MIB modules it supports, and the command-line utilities that are provided with it.	
	cffile.txt	Description of how to specify configuration settings for the agent by placing them in the sysedge.cf configuration text file.
	edgemon.txt	Description of how to use the edgemon command-line utility.
	edgewatch.txt	Description of how to use the edgewatch command-line utility.

Table 8. SystemEDGE Directories and Files for Windows Systems (Page 3 of 3)

Directory	File	Description
doc/ (continued)	emphistory.txt	Description of how to use the emphistory command-line utility.
	sendtrap.txt	Description of how to use the sendtrap command-line utility.
	sysvariable.txt	Description of how to use the sysvariable utility to query systems and exercise the agent.
	walktree.txt	Description of how to use the walktree utility to retrieve specific MIB variables.
	xtrapmon.txt	Description of how to use the xtrapmon utility to receive and log SNMP trap messages.

Directories and Files for All Platforms

Table 9 describes the directories and files that the SystemEDGE agent installation creates for all platforms.

Table 9. Directories and Files for All Platforms (Page 1 of 4)

Directory	File	Description
bin/	diagsysedge.exe	Diagnostic tool for automating the collection of troubleshooting information for the SystemEDGE agent.
config/	Subdirectory that contains the configuration files that the SystemEDGE agent reads on startup. To configure the agent for use in a local environment, copy these files to /etc (for UNIX) or to the system root directory (for Windows), and edit them to specify local values for items such as the following: license key for this system; communities, system contact person, system location, and other configurable parameters; and entries for the agent's self-monitoring and history sampling tables.	
	sysedge.lic	Text file that contains the license key that enables the agent to run on this system.
	sysedge.cf	Text file that contains local configuration parameters for MIB-II system group values, communities, self monitoring, and so on for the agent.

Table 9. Directories and Files for All Platforms (Page 2 of 4)

Directory	File	Description
config/ (continued)	sysedge.mon	Specially formatted text file specifying MIB objects and conditions that the agent will automatically monitor through its self-monitoring features.
	S99sysedge	(Solaris Only.) Shell script for starting and stopping the agent. If you disabled the native SNMP agent through the installation process, the installation script places this script in the /etc/rc2.d directory to start the agent automatically at system boot time.
	sysedge	(HP-UX Only.) Shell script for starting and stopping the agent. If you disabled the native SNMP agent through the installation process, the installation script places this script in /sbin/init.d and links /sbin/rc2.d/S990sysedge so that the SystemEDGE agent starts automatically at boot time. If you did not disable the native SNMP agent during the installation, you can do so by editing SnmpMaster, SnmpMib2, and SnmpHpunix in /sbin/init.d, and changing the following statements, as shown: <ul style="list-style-type: none"> • Change 'SNMP_MASTER_START=1' to 'SNMP_MASTER_START=0' • Change 'SNMP_HPUNIX_START=1' to 'SNMP_HPUNIX_START=0' • Change 'SNMP_MIB2_START=1' to 'SNMP_MIB2_START=0'
	sysedge	(Digital UNIX and Tru64 Only.) Shell script for starting and stopping the agent. If you disabled the native SNMP agent through the installation process, the installation script places this script in /sbin/init.d and links /etc/rc3.d/S99sysedge to it to start the agent automatically at boot time. If you did not disable the native SNMP agent during the installation, you can do so by editing snmpd in /sbin/init.d and putting an exit 0 statement before any of the snmpd script's code.

Table 9. Directories and Files for All Platforms (Page 3 of 4)

Directory	File	Description
config/ (continued)	sysedge	(Linux Only.) Shell script for starting and stopping the agent. If you disabled the native SNMP agent through the installation process, the installation script places this script in /etc/rc.d/init.d and link /etc/rc.d/rc2.d/S99sysedge to it to start the agent automatically at boot time. If you did not disable the native SNMP agent during the installation, you can do so by editing snmpd in /etc/init.d, and putting an exit 0 statement before any of the snmpd script's code.
contrib/	Subdirectory that contains the SystemEDGE agent dynamic link library, command-line utilities for use with SystemEDGE self monitoring, and sample extension and action scripts.	
	auto.install	Sample scripts which automate the installation, deployment, and licensing of the SystemEDGE agent distribution.
	call-sendtrap.c	Sample C code to call the sendtrap utility from within a program.
	checkprinter.exe	Sample extension script for checking the printer status on Windows systems.
	countmail.sh	Sample script that measures the mail message throughput of the SMTP (Sendmail) service.
	getextension.sh	Sample multi-purpose extension script.
	logfileaction.sh	Sample log file action script for sending alternative trap formats.
	mailaction.sh	Sample mail action script for sending a mail message with variable bindings.
	mqcnt	Sample extension script for measuring Sendmail queue length.
	ntdist.pl	Sample script for deploying SystemEDGE agents and eHealth AIMs.
	nteventfilter.pl	Sample script for filtering NT events and sending traps that look like they are originating from the SystemEDGE agent.

Table 9. Directories and Files for All Platforms (Page 4 of 4)

Directory	File	Description
contrib/ (continued)	ping.sh	Sample extension script for performing remote ping operations.
	restartsvc.exe	Sample action script for restarting a Windows service.
	sendpage.pl	Sample script for paging a PageMart alphanumeric pager.
	smtp-listenq.sh	Sample script that measures the listen backlog of the SMTP port and reports its current length.
doc/	Subdirectory that contains documentation for the SystemEDGE agent, the MIB modules it supports, and the command-line utilities that are provided with it.	
	empire.asn1	Systems Management MIB specification. This document describes the meaning and use of all of the management objects in the Systems Management MIB.
	hostmib.asn1	Host Resources MIB specification (RFC 1514). This document describes the meaning and use of all of the management objects in the Host Resources MIB.
	sysedge.pdf	PDF copy of the <i>eHealth SystemEDGE User Guide</i> .
	relnotes.pdf	PDF copy of the <i>eHealth SystemEDGE Release Notes</i> .

NOTE

The SystemEDGE agent installation script enables you to install SystemEDGE as your default SNMP agent, to configure SystemEDGE to start up by default when the system is booting, to turn off your native SNMP agent, and to make changes to the startup (rcX.d files). If you did not configure the agent to start automatically during the installation, you can change those settings manually.

Uninstalling the SystemEDGE Agent

This section explains how to remove the files and subdirectories that are associated with the SystemEDGE agent.

NOTE

The following steps also remove the agent's license file (sysedge.lic), configuration file (sysedge.cf), and Monitor table configuration file (sysedge.mon) from the /etc (UNIX) or %SystemRoot%\system32\ (Windows) directory. *If you want to save these files, copy them to another directory before you run the Remove utility.*

Uninstalling SystemEDGE for Solaris Systems

Use the Solaris package-remove utility (pkgrm) to remove the SystemEDGE agent package (EMPsysedg) from your system:

1. Run the pkgrm utility by entering the following command:

```
pkgrm EMPsysedg
```

2. Answer **y** (yes) to the prompts.

Uninstalling SystemEDGE for Windows Systems

The uninstall program removes SystemEDGE. It also removes the sysedge.dll, sysedge.cf, sysedge.lic, and sysedge.mon files from the %SystemRoot%\system32\ directory.

NOTE

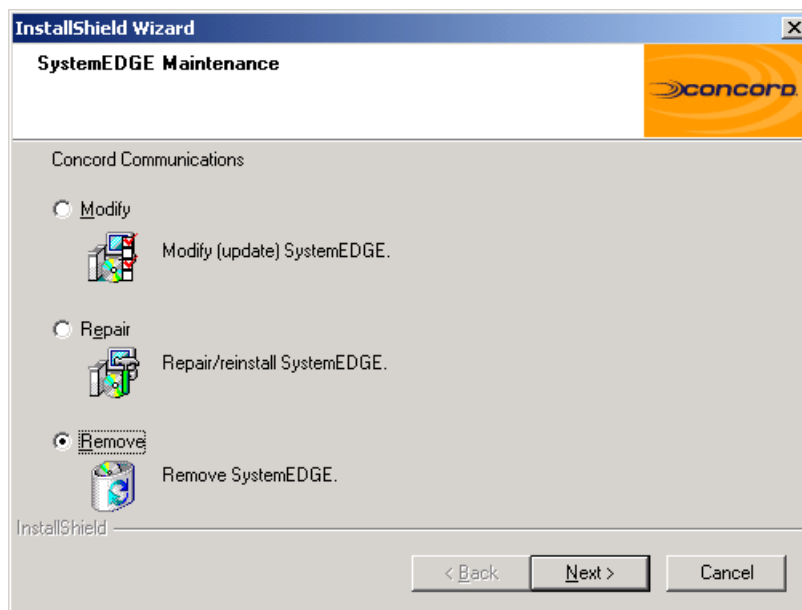
If you want to save these files, copy them to another directory before you run the uninstall utility.

If you installed the agent with the InstallShield installer, refer to the next section. If you installed the agent from the command line, refer to “Uninstalling an Agent from the Command Line” on page 101.

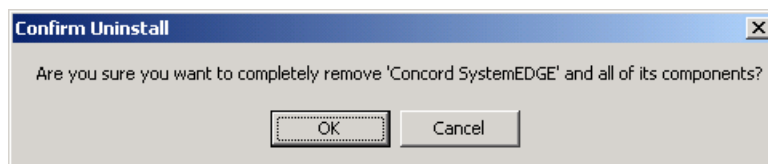
Uninstalling an Agent with InstallShield

If you used InstallShield to install the agent, you can also use it to remove the agent:

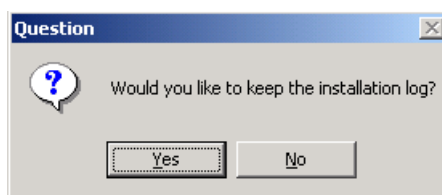
1. Log on to the Windows system as **administrator**.
2. Insert the CD that contains the software distributions into the CD-ROM drive. Windows automatically mounts the drive.
3. Double-click sysedge.exe from the CD-ROM to run the self-extracting executable. The SystemEDGE Maintenance dialog box appears.



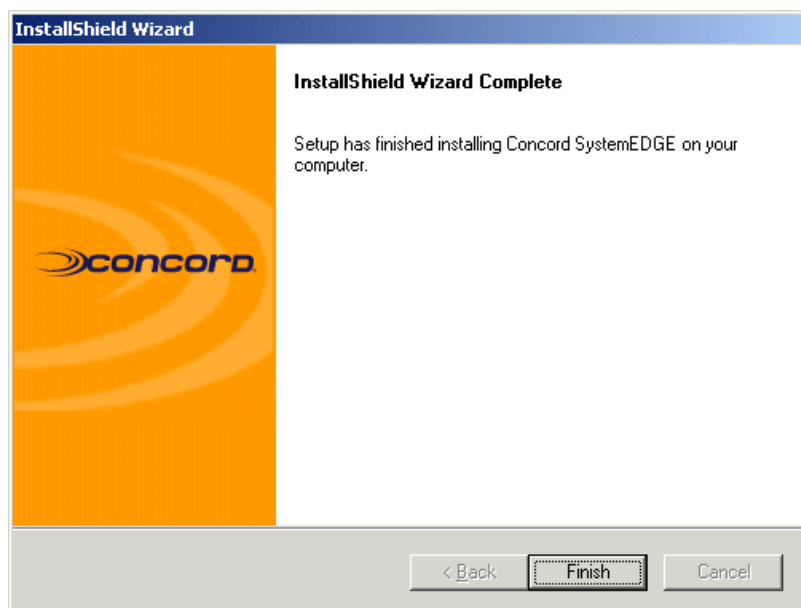
4. Select **Remove** and click **Next**. The following prompt appears.



5. Click **OK**. The following prompt appears.



6. Click **Yes** to save a copy of the log file, or **No** if you do not want to save the log file. The following dialog box appears.



7. Click **Finish**.

Uninstalling an Agent from the Command Line

If you installed the agent from the command line, you can remove it by running the setup utility with the `-r` argument. The `-r` arguments instructs the setup utility to remove SystemEDGE. It also removes the `sysedge.dll`, `sysedge.cf`, `sysedge.lic`, and `sysedge.mon` files from the `%SystemRoot%\system32` directory.

NOTE

If you want to save these files, copy them to another directory before you run the SystemEDGE agent setup utility.

To remove the command-line installation of the SystemEDGE agent for Windows:

1. Log on to the Windows system as administrator.
2. Change to the SystemEDGE installation directory by entering the following, where `C:\sysedge` is the directory where you installed the agent:

```
C:\sysedge
```

3. Enter the following to remove the software:

```
setup -r
```

Uninstalling SystemEDGE for HP-UX Systems

Use the HP-UX `swremove` utility to remove the SystemEDGE agent software package (EMPsysedge). Run the `swremove` utility by entering the following:

```
swremove -v EMPsysedge
```

Uninstalling SystemEDGE for Linux Systems

The SystemEDGE agent home directory contains a removal shell script (Remove) that removes all subdirectories and files under the SystemEDGE agent home directory but does not remove the home directory or the Remove script itself.

1. Run the Remove script by entering the following:

```
./Remove
```

2. Remove the agent's configuration files by entering the following commands, one at a time:

```
rm /etc/sysedge.lic
```

```
rm /etc/sysedge.cf
```

```
rm /etc/sysedge.mon
```

Uninstalling SystemEDGE for AIX Systems

Use the AIX smit utility to remove the SystemEDGE agent software package (EMPsysedge).

1. Run the smit utility by entering the following:

```
smit install_remove
```

2. Select **EMPsysedge.rte** from the list of software to remove.

3. Remove the Remove script and the SystemEDGE agent's home directory (typically named sysedge) by entering the following commands, one at a time:

```
rm Remove
```

```
cd ..
```

```
rmdir agent-homedir
```

Uninstalling SystemEDGE for Digital UNIX and Tru64 Systems

Use the Digital UNIX or Tru64 setld utility to remove the SystemEDGE agent software package (EMPsysedge). Run the setld utility by entering the following:

```
setld -d EMPSYSEDGE
```


Licensing the SystemEDGE Agent

The SystemEDGE agent utilizes a host-based license method, which means that you can run copies of the agent only on systems that possess a valid license key. This chapter provides an overview of the licensing options and explains how to obtain a license key.

NOTE

You can license the agent during the installation procedure, as well as during agent deployment if you are deploying the agent from a Windows system through AdvantEDGE View. For more information about deploying and licensing the agent, refer to Chapter 18, “Deploying the SystemEDGE Agent.” This chapter describes alternative methods of licensing.

Licensing the Agent

The SystemEDGE agent displays a message that it did not find a valid license when it starts, unless you licensed the agent during the installation process. Upon startup, the unlicensed agent provides you with a **public key** that you must send to CA. CA uses that public key to generate a valid license key for your system.

An unlicensed SystemEDGE agent starts in **restricted mode**. In this mode, the agent can respond to a limited group of SNMP variables to support remote licensing. For example, you can query the agent for its version and licensing information. If you configured a read-write community string for the agent, you can license it remotely using an SNMP Set command.

License Keys

The SystemEDGE agent can run on a system that includes a license key that is valid for that system. The license key must exist in the `sysedge.lic` file in the `/etc` directory for UNIX systems or the `%SystemRoot%\system32` directory for Windows systems. A license key consists of four space-separated, 8-character sequences, such as the following:

```
YFABHSgN cSzdjW92 nxXjO05w tHYBXs9N
```

The following is a sample SystemEDGE agent license file. (A pound sign (#) character in column 1 indicates that the entire line is a comment.)

```
# license file for SystemEDGE Agent
# Concord Communications, Inc.
# http://license.concord.com
#
# file /etc/sysedge.lic or
# %SystemRoot%\system32\sysedge.lic
# A valid license key has four parts of 8 characters
# per part. parts are separated by space(s) with one
# license key per line

# sysedge jupiter sol2 5.9 807cb1da007cb1da 4.2 PL 1
e13311d3 0F2a7cb1 abC512dc fF8C923a
```

NOTE

You must license both permanent and evaluation copies of the agent. If you are evaluating the SystemEDGE agent, CA can provide you with a temporary license that allows the agent to operate during the evaluation period.

Licensing Methods

Table 10 lists the methods for licensing the SystemEDGE agent and identifies which are useful for local and remote licensing, which provide automatic licensing, which provide proxy support, and which you can use even if the system that you are licensing does not have internet access.

Table 10. Licensing Methods

Licensing Method	Local?	Remote?	Automatic?	Proxy Support?	Possible without Internet Access for the Target? ¹
Licenseeme utility	Yes	Yes	No	Yes	Yes
AdvantEDGE View Host Administration	Yes	Yes	No	Yes	Yes
AdvantEDGE View Event Processing	Yes	Yes	No	Yes	Yes
AdvantEDGE View Agent Deployment	Yes	Yes	Yes	Yes	Yes
eHealth (TrapEXPLODER)	Yes	Yes	Yes	Yes	Yes
Web-based form	Yes	Yes	No	Yes	Yes
Licenseutil.pl script	Yes	Yes	Yes ²	Yes	Yes
1. The system from which the licensing tool is running <i>does</i> require Internet access, but the target system does not. 2. Use a scheduler daemon like cron to run the licenseutil.pl script and license systems on your network automatically.					

Obtaining a License Key

The SystemEDGE agent installation provides the option for you to license the agent automatically through the `licenseme` utility. If you did not license the agent during the installation process, you can obtain a license key manually, as described in the following section.

To obtain a license key:

1. Start the agent by entering one of the following commands from the `sysedge` directory at the command prompt.

For UNIX systems, enter the following:

```
bin/sysedge
```

For Windows systems, enter the following:

```
setup -l
```

NOTE

You must be logged in as root or an administrator to start the agent. The agent initially starts in unlicensed, restricted mode.

The setup program displays a message similar to the following:

```
SystemEDGE Version 4.2 Patch level 1
Copyright 2004 by Concord Communications, Inc.
Could not find a valid license for machine 'pluto'
http://license.concord.com/
Provide the following: sysedge pluto sol2 5.9 8035b1f8f643ab43 4.2 PL 1
```

You must provide the public key (sysedge pluto sol2 5.9 8035b1f8f643ab43 4.2 PL 1 in this example) to CA to receive your license key.

2. Obtain your license key through one of the methods described in the following sections:
 - Run the `licenseme` utility. For more information, refer to the next section, “Running the `licenseme` Utility.”
 - Complete the Web-based licensing form through the Internet. For more information, refer to “Using the Web-Based Licensing Form” on page 111.
 - Send an e-mail request to `productlicensing@ca.com`, and place the returned license key in the appropriate license file. For more information, refer to “Sending an E-mail Request for a License Key” on page 110.
 - Run the CA-supplied `licenseutil.pl` script. For more information, refer to “Using the `licenseutil.pl` Utility” on page 112.
 - Use AdvantEDGE View Agent Deployment to deploy and license your agents automatically. For more information, refer to Chapter 18, “Deploying the SystemEDGE Agent.”

Running the `licenseme` Utility

You can obtain a license for the SystemEDGE agent by running the `licenseme` utility from the `/bin` subdirectory of the SystemEDGE agent distribution. This utility automates the interaction with the license Web server and therefore requires Web access (either directly or through a Web proxy) from the system that is running the utility. It also requires SNMP Set permission to the target agent. It uses SNMP to query the license information from the target agent, automatically fills out the online Web-based licensing form, retrieves the resulting license key, and uses an SNMP Set to configure that key to the target agent. The target agent then saves the license key to its license file. You can run `licenseme` interactively or invoke it with scripts.

To run the licenseme utility:

1. Change to the directory where you installed the SystemEDGE agent (for example, /opt/EMPsysedge or C:\sysedge).
2. Enter the following at the command prompt:

```
bin/licenseme
```

The licenseme utility prompts you for your account information on the licensing Web site (<http://license.concord.com>). Once you have provided this information, the program contacts the Web site and retrieves the license for you. If you have a valid read-write community string configured, licenseme saves the license for you and automatically enables full SystemEDGE agent operation. For information about configuring a read-write community string, refer to Chapter 4, “Configuring the SystemEDGE Agent.”

NOTE

You can use the licenseme program to license SystemEDGE agents that are installed on remote systems. If you have only one system that has access to the Web, you can run licenseme on that system and direct it to license your other systems.

Sending an E-mail Request for a License Key

You can send an e-mail request to productlicensing@ca.com to obtain a license key. Include your customer ID and name in all e-mail requests. CA returns the license key within approximately one business day of the request. When you receive the license key through e-mail, place it in your sysedge.lic file.

NOTE

You do not need direct Internet access or SNMP Set permissions to use this method of obtaining a license key.

Using the Web-Based Licensing Form

If automated licensing is not available, fill out the online Web-based license form that is available at the following URL: <http://license.concord.com>.

1. Using a Web browser, go to the Product licensing Web site at <http://license.concord.com>, and select the **Create License** option that matches your use of the agent.

NOTE

You must supply a user name and password to access the license form.

2. Complete the license form, entering the public key that was provided by the SystemEDGE agent. You must supply the following information:
 - Name
 - E-mail address
 - SystemEDGE agent release number and patch level
 - System name
 - Operating system name and version
 - System identifier

After you submit the license request form, the Web server generates a license and displays it in your Web browser. It also e-mails the license to the contact person in your organization.

3. Copy the license into the `sysedge.lic` file. This file is located in the `/etc` or `%SystemRoot%\system32` directory.

The license key is case sensitive. Copy it exactly as it appears. If possible, use your system's copy-and-paste facility instead of typing it by hand. If you are entering the license key manually, be careful not to confuse characters such as the letters **l** and **1** and the number **1**, or the letter **O** and the number **0**.

4. Save the sysedge.lic file, and then restart the SystemEDGE agent.

For UNIX systems, restart the SystemEDGE agent by entering the following from the /opt/EMPsysedge directory when you are logged in as root:

```
bin/sysedge -b
```

For Windows systems, do one of the following:

- Enter the following commands from the command line:

```
net stop sysedge  
net start sysedge
```

- Use the SystemEDGE Control Panel by selecting **Start** → **Settings** → **Control Panel** → **eHealth SystemEDGE**. On the SystemEDGE Control Panel, click **Stop Agent** and then **Start Agent**.

The SystemEDGE agent is now licensed and ready to use.

Using the licenseutil.pl Utility

The licenseutil Perl script parses a list of hostnames that you want to license, collects license information from each system, and then forwards the list of license keys to CA. CA returns the list of licenses to the script, and then the script populates each system's sysedge.lic file with those licenses using an SNMP Set. The script requires a freely available Perl LWP module for WWW manipulation.

Before You Run `licenseutil.pl`:

1. Install the SystemEDGE agent and any plug-in modules.
2. Start the SystemEDGE agent. (It starts in restricted mode.)
3. Determine whether you want to run `licenseutil.pl` from a system that is connected to the Internet or from a system that is not connected to the Internet.

Using `licenseutil.pl` from a System that is Connected to the Internet.

1. Ensure that the system on which you want to run `licenseutil.pl` has connectivity to all of the systems that require licenses, as well as to the Internet.
2. Change to the `/sysedge/contrib` directory and unzip the `licensetools` file. The distribution includes `licenseutil.pl`, sample host files and license files, and Perl modules for Perl/SNMP queries.
3. Open the `licenseutil.pl` file in a text editor, and provide values for the following variables:
 - a. Set `$username` to your licensing user name.
 - b. Set `$passwd` to your licensing password.
 - c. Set `$enduser` to the name of the end-user company, if you are requesting a usage-based license.
 - d. Set `$name` to your name.
 - e. Set `$email` to your e-mail address.
 - f. If the type of license that you are selecting is anything other than permanent, set `$licensetype` to one of the other values (partner, evaluation, or usage-based).
 - g. If you are requesting a usage-based license, set `$duration` to the number of months for which the license is valid.
 - h. Set the values for `$proxy`, `$proxyport`, `$proxyuser`, and `$proxypasswd` if you are using an authenticated proxy to access the Internet.

- i. In the section, “Comment out plugins which are not desired,” place a pound sign (#) in front of the modules for which you do not want to obtain licenses. Modules are named as follows:
 - sysedge is the SystemEDGE agent.
 - svcrsp is eHealth Service Availability.
 - xchgmod is eHealth AIM for Microsoft Exchange.
 - iismod is eHealth AIM for Microsoft IIS.
 - apachemod is eHealth AIM for Apache.
 - oramod is eHealth AIM for Oracle.
 - sqlmod is eHealth AIM for Microsoft SQL Server.
 - netsvcmmod is eHealth AIM for Network Services.
 - firewall is eHealth AIM for Check Point FireWall-1.
 - ccmmod is the eHealth for Cisco CallManager module.
 - mosmod is the eHealth Voice Quality Monitor module.
 - j. Set \$OS to either UNIX or Windows, depending on the type of operating system that you are using.
4. Save and close licenseutil.pl.
 5. Open the hostlist file in a text editor, edit it to include the hostnames or IP addresses of the systems to license, and then save and close the file.
 6. Run licenseutil.pl by entering the following at the command line:

```
licenseutil.pl -auto hostfile community port timeout [product]
```

When you execute licenseutil.pl with the -auto option, it collects information from all systems in the host list, interacts with the CA licensing server to collect the license key information, and then populates the sysedge.lic file on each of the systems through an SNMP Set command.

Using `licenseutil.pl` from a System that is Not Connected to the Internet.

1. Ensure that the system on which you are running `licenseutil.pl` has connectivity to the systems that require licenses.
2. Change to the `drive:/sysedge/contrib` directory and unzip the `licensetools` file. The distribution includes `licenseutil.pl`, sample host files and license files, and Perl modules for Perl/SNMP queries.
3. Edit the `licenseutil.pl` and `hostlist` files as described in “Using `licenseutil.pl` from a System that is Connected to the Internet” on page 113.
4. Run `licenseutil.pl` by entering the following at a command line, substituting the product string for `product`, as described in “Using `licenseutil.pl` from a System that is Connected to the Internet” on page 113:

```
licenseutil.pl -get hostfile product community port timeout
```

5. Send an e-mail to `productlicensing@ca.com` that includes the license file returned by the script and your user name and customer ID. CA sends a license file in return.
6. Run `licenseutil.pl` again using the `-set` option, as follows:

```
licenseutil.pl -set licensefile community port timeout
```

The script performs an SNMP Set to each agent that is defined in your `hostlist` file, populating its `sysedge.lic` file, and each agent becomes fully licensed.

NOTE

You can run the `licenseutil.pl` script interactively or with the `cron` command. If you use `cron`, you must update the `hostlist` file, but you do not need to perform any other tasks. Agents that are running in restricted mode will be licensed and will enter full mode. Agents that are already licensed are not affected by the script. They will continue to run as before.

Configuring the SystemEDGE Agent

This chapter describes how to set configuration parameters that the SystemEDGE agent reads on startup. These configuration parameters are defined in the `sysedge.cf` text file, and they specify local system values such as the following:

- System description, community strings, and trap communities
- Agent behavior for reporting or not reporting security-related information and running or not running action scripts
- Entries for the agent's self-monitoring tables

The `sysedge.cf` file is located in the `/etc` (UNIX) or `%SystemRoot%\system32` (Windows) directory. For Windows systems, you can also access `sysedge.cf` through the SystemEDGE Control Panel by selecting **Start** → **Settings** → **Control Panel** → **eHealth SystemEDGE**, and then clicking `sysedge.cf` on the SystemEDGE Control Panel.

NOTE

Whenever you modify the `sysedge.cf` file, you must stop and restart the SystemEDGE agent for your changes to take effect.

Interactions Between `sysedge.cf` and `sysedge.mon`

The SystemEDGE agent uses the `sysedge.mon` file as a backing store for the agent's self-monitoring tables. The `sysedge.cf` and `sysedge.mon` files interact, and entries in `sysedge.cf` file take precedence over entries in `sysedge.mon`. The `sysedge.cf` file is static and cannot be edited remotely. The `sysedge.mon` file is not static, so AdvantEDGE View and other management software can update this file through SNMP Sets.

If you remove a monitoring entry from `sysedge.cf` and that entry also exists in `sysedge.mon`, you must also remove it from `sysedge.mon` to prevent the agent from using it. When you add configuration entries to the agent through AdvantEDGE View, the entries are stored in `sysedge.mon` and are *not* added to `sysedge.cf`. For more information about using `sysedge.mon`, refer to Appendix C.

Configuring the Agent During the Installation Procedure

The SystemEDGE installation enables you to perform some configuration tasks when you install the agent. For example, you can configure system description and location, read-only and read-write communities, and trap destinations during the installation process. You can later modify any of those settings by editing the `sysedge.cf` file, as described in this chapter.

Before You Begin

Make sure that your installation has copied the `sysedge.cf` file to the `/etc` (UNIX) or `%SystemRoot%\system32` (Windows) directory. Keep in mind that if you performed an upgrade instead of a clean installation, the installation does not overwrite your existing `sysedge.cf`, `sysedge.lic`, or `sysedge.mon` files. You can compare the existing versions with the new versions, which are installed in the `config` subdirectories, modify the new versions, save any information that you want to keep from the existing files, and then copy the new files into the `/etc` (for UNIX) and `%SystemRoot%\system32` (for Windows) directories.

To copy the file manually:

For UNIX systems, enter the following:

```
cp config/sysedge.cf /etc
```

For Windows systems, enter the following:

```
copy config\sysedge.cf %SystemRoot%\system32\
```

Sample sysedge.cf File

The following is a sample sysedge.cf file:

```
# Concord SystemEDGE configuration file
# Copyright 2004 Concord Communications, Inc.
# http://www.concord.com/support
# http://www.concord.com/sysedge
# http://www.concord.com/sysedge-contrib
#
# Tell the agent that we do not want to support Empire's
# User and Group tables due to security policies
# Default is to support those groups
#
# no_usergroup_table
#
# Tell the agent not to support its remoteShell MIB object group
# due to security policies
# default is to support this group
#
# no_remoteshell_group
#
# Tell the agent not to query the status of serial devices
# other than the keyboard and mouse; the Agent will return
# unknown(1) when queried for a serialport status if the command
# below is uncommented; the default is to support serial port status
# queries
#
# no_serial_status
#
# Tell the agent not to query the status of floppy drives.
#
# no_stat_floppy
```

```
#
# Tell the agent not to allow SNMP Sets in the Empire process or
# HostMIB runningSoftware tables. The default is to allow SNMP
# Sets to those tables given a valid community string supporting
# read-write access. Allowing Sets to these two tables allows
# remote managers to send processes signals and kill them if
# desired.
#
# no_process_sets
#
# Tell the agent not to support Actions. The default is to invoke
# Actions when a monitor, log-monitor, or event-monitor
# table entry evaluates to True. Actions are executed
# as the root so care must be taken as to which Action scripts
# are invoked.
#
# no_actions
#
# Tell the agent not to support the Installed Software table of
# the Host Resources MIB. (This token is currently valid only on Linux
# systems.)
# no_hrsinstalled
#
#
# You can extend the Empire Enterprise MIB by adding new
# scalar variables. Each entry here represents a new leaf in the
# Extension group of the Empire Enterprise MIB.
# The values for these variables are determined
# by executing a command on the host where the agent is running.
# See the SystemEDGE Agent User Manual for more information.
#
# Included here are some sample MIB variables that match the sample
# entries in the Empire Enterprise MIB. To use these examples you
# should specify the path to the getextension program and uncomment
# the following lines.
#
# extension 1 OctetString ReadOnly \sysedge\contrib\getextension.bat
# extension 2 OctetString ReadOnly \sysedge\contrib\getextension.bat
# extension 3 OctetString ReadOnly \sysedge\contrib\getextension.bat
# extension 4 OctetString ReadOnly 'C:\sysedge\bin\getver.exe -f
# %SystemRoot%\system32\sysedge.dll'
# extension 5 OctetString ReadWrite 'C:\sysedge\bin\bounce.exe'
```



```

# extension 10 Integer ReadOnly 'C:\sysedge\bin\checkfile.exe C:\root.exe'
#
# In Windows systems you can further extend the Empire Enterprise MIB
# by adding new variables from the NT Registry and Perfmon services.
#
# These values are obtained directly from the system without the need
# for an external program.
# See the SystemEDGE Agent User Manual for more information.
#
# Included here are some sample MIB variables that user this facility.
#
# ntregperf 1 OctetString Registry
# 'SYSTEM\CurrentControlSet\Control\CrashControl' 'DumpFile'
# ntregperf 2 Gauge Performance 'Objects' 'Threads' '1'
# ntregperf 3 Counter Performance 'TCP' 'Segments Sent/sec' '1'
# ntregperf 4 Integer Registry 'SYSTEM\CurrentControlSet\Services\SNMP' 'Start'
# ntregperf 5 Integer Registry 'HARDWARE\DESCRIPTION\System' 'Component
# Information'
# ntregperf 6 OctetString Registry 'HARDWARE\DESCRIPTION\System'
# 'SystemBiosVersion'
# ntregperf 7 Integer Performance 'Memory' 'Available Bytes' '1'
# ntregperf 8 Counter Performance 'TCP' 'Segments Received/sec' '1'
# ntregperf 9 Counter Performance 'UDP' 'Datagrams Received/sec' '1'
# ntregperf 10 Counter Performance 'UDP' 'Datagrams Sent/sec' '1'
# ntregperf 11 Integer Performance 'System' 'System Up Time' '1'
# ntregperf 12 Gauge Performance 'Objects' 'Processes' '1'
#
# NT Event monitoring entries are defined below; each line
# instructs the agent to monitor a given NT Event log for a given
# rule match including regular expressions.
# (see man ed(1) for more details on regular expression syntax)
#
# Usage:
# watch ntevent index flags Eventlog Type 'Source-Match' 'Description-Match'
# 'Description' 'Action'
#
# Examples:
# watch ntevent 1 0x8 System Error '.*' '.*' 'System Error - INFO' ''
# watch ntevent 2 0x8 Security Fail '.*' '.*' '' ''
# watch ntevent 3 0x8 Security All '.*' 'EMP' 'Security - WARNING' ''

```

```
# watch ntevent 100 0x8 System All 'W3SVC' '.*Logon.*failure.*' 'Web Logon
# Failure' ''
# watch ntevent 101 0x8 System All 'MSFTPSVC' '.*Logon.*failure.*' 'FTP Logon
# Failure' ''
# watch ntevent 102 0x8 System All 'MSFTPSVC' '.*User.*timed-out.*' 'FTP User
# Time-out' ''
# watch ntevent 120 0x8 Application Error '.*' '\[5\]' 'Error: File, Unexpected
# error' ''
# watch ntevent 122 0x8 Application All '.*' '\[4101\]' 'An object call caused
# an exception' ''
# watch ntevent 123 0x8 Application All '.*' '\[4131\]' 'A server process
# failed during initialization' ''
# watch ntevent 124 0x148 Application All '.*' '\[4153\]' 'An exception
# occurred within a Resource Dispenser' ''
# watch ntevent 125 0x148 Application All '.*' '\[4180\]' 'MTS recorder is out
# of memory. Unable to allocate new resources' ''
# watch ntevent 126 0x148 Application All 'Failover' '\[1\]' 'The following IP
# address was added to target monitoring list:' ''
# watch ntevent 127 0x148 Application All 'Failover' '\[2\]' 'The following IP
# address was added to target monitoring list:' ''
# watch ntevent 128 0x148 Application All 'Failover' '\[4\]' 'Event ID 4' ''
# watch ntevent 129 0x148 Application All 'Failover' '\[5\]' 'Event ID 5' ''
# watch ntevent 130 0x148 System All '.*' '\[301\]' 'Out of Virtual Memory.
# Please close some applications.' ''
# watch ntevent 131 0x148 System All '.*' '\[10010\]' 'The server did not
# register with DCOM within the required time out' ''
# watch ntevent 133 0x148 System All '.*' '\[14\]' 'The HTTP Filter DLL failed
# to load. The data is the error' ''
# watch ntevent 135 0x148 System All '.*' '\[19\]' 'The HTTP server encountered
# an unhandled exception while processing the ISAPI Application.' ''
# watch ntevent 136 0x148 System All '.*' '\[29\]' 'The server failed to
# shutdown application.' ''
# watch ntevent 137 0x148 System All '.*' '\[37\]' 'Out of process application;
# terminated unexpectedly' ''
# watch ntevent 167 0x8 System Error 'EDB' '.*' 'Monitor MSSQL Server Database
# Events' ''
# watch ntevent 168 0x8 Security Error 'EDB' '.*' 'Monitor MSSQL Server
# Database Events' ''
```

```

# watch ntevent 169 0x8 Application Error 'EDB' '.*' 'Monitor MSSQL Server
# Database Events' ''
# watch ntevent 170 0x8 Application All '.*' '\[1\]' 'Event ID 1' ''
# watch ntevent 171 0x8 Application All '.*' '\[2\]' 'Event ID 2' ''
# watch ntevent 172 0x8 Application All '.*' '\[4\]' 'Event ID 4' ''
# watch ntevent 173 0x8 Application All '.*' '\[5\]' 'Event ID 5' ''
#
# Log file monitoring entries are defined below; each line
# instructs the agent to watch a given log file for a given
# regular expression (see man ed(1) for more details on regular
# expression syntax)
#
# Usage:
# watch logfile index flags logfilename 'regexpr' 'description' 'action'
#
# Examples:
#   Note, these are examples from Unix systems. NT does not use
#   text logfiles to record operating system messages. You could
#   use this however for application logfiles.
#
# watch logfile 1 0x8 /var/adm/messages 'su.*fail' 'su attempt - WARNING' ''
# watch logfile 2 0x8 /var/adm/messages 'sysedge.*fail' 'sysedge fail -
# NOTICE' ''
# watch logfile 3 0x8 /usr/adm/sulog 'su.* - ' 'su attempt - WARNING' ''
# watch logfile 4 0x8 /var/log/daemon-log 'su.*fail' 'su attempt - WARNING' ''
# watch logfile 10 0x8 C:\eHealth\log\pollerStatus\messages.stats.log 'error'
# 'eH poller error' ''

# You can add process monitoring entries to have the agent automatically
# monitor a running process. This creates an entry in the Empire Process
# Monitor Table.
#
# Usage:
# watch process procAlive 'regexpr' index flags interval 'descr' 'action'
# watch process attribute 'regexpr' index flags interval stype oper value
# 'descr' 'action'
#

```

```
# Examples:
# watch process procAlive 'netscape|NETSCAPE' 1 0x8 60 'Netscape Alive'
# 'C:\sysedge\contrib\restartproc.bat'
# watch process procRSS 'netscape|NETSCAPE' 2 0x200508 60 absolute > 50000
# 'Netscape Size' ''
# watch process procAlive 'nhiServer' 50 0x8 30 'eH Nhi Server' ''
# watch process procAlive 'nhiMsgServer' 51 0x8 30 'eH Message Server' ''
# watch process procAlive 'nhiLiveExSvr' 52 0x8 30 'eH LiveEx Server' ''
# watch process procAlive 'nhiDbServer' 53 0x8 30 'eH Db Server' ''
# watch process procAlive 'nhiCfgServer' 54 0x8 30 'eH Cfg Server' ''
# watch process procAlive 'nhihttpd' 55 0x8 30 'eH Httpd Server' ''
# watch process procAlive 'nhiPoller' 56 0x8 30 'eH Poller' ''
# watch process procAlive 'inetinfo|INETINFO' 400 0x8 30 'MS IIS' ''
# watch process procAlive 'dsamain|DSAMAIN' 401 0x8 60 'MS Exchange Directory
# Service' ''
# watch process procAlive 'emsmta|EMSMTA' 402 0x8 60 'MS Exchange MTA' ''
# watch process procAlive 'imc|IMC' 403 0x8 60 'MS Exchange Internet Mail
# Connection' ''
# watch process procAlive 'store|STORE' 404 0x8 60 'MS Exchange Info Store' ''
# watch process procAlive 'mad|MAD' 405 0x8 30 'MS Exchange Attendent'
# watch process procAlive 'events|EVENTS' 406 0x8 30 'MS Exchange Event
# Service'
# watch process procAlive 'sqlservr|SQLSERVER' 407 0x8 30 'MS SQL Server' ''
# watch process procAlive 'sqlexec|SQLEEXEC' 408 0x8 60 'SQL Executive Service'
# ''
# watch process procAlive 'services|SERVICES' 409 0x8 60
# 'Alerter/Browser/Messenger/EventLog Service' ''
# watch process procAlive 'dbabrd|DBABRD' 412 0x8 60 'Backup Agent
# Communication Server' ''
# watch process procAlive 'dbasvr|DBASVR' 413 0x8 60 'Backup Agent RPC Server'
# ''
# watch process procAlive 'msdtc|MSDTC' 414 0x8 60 'MSDTC Service' ''
# watch process procAlive 'lsass|LSASS' 415 0x8 60 'Net Logon' ''
# watch process procAlive 'RpcSs|RPCSS' 416 0x8 60 'Remote Procedure Call (RPC)
# Service' ''
```

```
# watch process procAlive 'pcmsvc32|PCMSVC32' 417 0x8 60 'SMS Package Command
# Manager NT' ''
# watch process procAlive 'wuser32|WUSER32' 418 0x8 60 'SMS Remote Control
# Agent' ''
# watch process procAlive 'timeserv|TIMESERV' 419 0x8 60 'Time Service' ''
# watch process procAlive 'wins|WINS' 420 0x8 60 'WINS service' ''
# watch process procAlive 'dns|DNS' 421 0x8 60 'WINS/DHCP' ''
# watch process procAlive 'mqsvc|MQSVC' 438 0x8 60 'MS Message Queue Service'
# ''
# watch process procAlive 'clcfg|CLICFG' 443 0x8 60 'SMS Client Config
# Manager' ''
# watch process procAlive 'smsexec|SMSEXEC' 444 0x8 60 'SMS Executive' ''
# watch process procAlive 'preinst|PREINST' 445 0x8 60 'SMS Hierarchy Manager'
# ''
# watch process procAlive 'invwin32|INVWIN32' 446 0x8 60 'SMS Inventory Agent
# NT' ''
# watch process procAlive 'siteins|SITEINS' 447 0x8 60 'SMS Site Config
# Manager' ''
# watch process procAlive 'sqlservr|SQLSERVER' 448 0x8 60 'SQL Server' ''
# watch process procAlive 'sqlexec|SQLEEXEC' 449 0x8 60 'SQL Executive' ''
# watch process procAlive 'db2serv|DB2SERV' 452 0x8 60 'Monitor db2serv
# Service' ''
# watch process procAlive 'tn3servr|TN3SERVER' 456 0x8 60 'Monitor TN3270
# Service' ''
# watch process procAlive 'iexplore|IEXPLORE' 457 0x8 60 'IE Alive'
# 'C:\sysedge\contrib\restartproc.bat'
# watch process procRSS 'iexplore|IEXPLORE' 458 0x200508 60 absolute > 50000
# 'IE RSS Size' ''
#
# NT service monitoring examples
# watch ntservice 'Simple TCP/IP Services' 10 0x8 60 'Monitor TCP services' ''
# watch ntservice 'World Wide Web Publishing Service' 500 0x8 30 'IIS WWW
# Service' ''
# watch ntservice 'Exchange.*Store' 501 0x8 30 'Exchange Info Store'
# watch ntservice 'sqlagent' 502 0x8 30 'SQL Server Agent Service' ''
#
```

```
# You can add process group monitoring entries to have the agent automatically
# monitor a group of running processes.
#
# Usage:
# watch procgroup 'regexp' index flags interval 'descr' 'action'
#
# Examples:
# watch procgroup 'dsamain|emsmta|store|mad|DSAMAIN|EMSMTA|STORE|MAD' 01 0x0 60
# 'Watch Exchange' ''
#
# You can add self-monitoring entries to have the agent automatically
# monitor MIB variables. This creates an entry in the Empire Monitor Table.
#
# Usage:
# monitor oid objid/name index flags interval stype oper value 'descr' 'action'
#
# Examples:
# monitor oid numInterrupts.0 14 0x200508 60 delta > 10000 'Interrupt Rate' ''
# monitor oid numPageFaults.0 15 0x200508 60 delta > 1000 'Page-fault Rate' ''
# monitor oid hrSystemProcesses.0 19 0x200508 60 absolute > 120 'Num Processes'
# '
# monitor oid loadAverage1Min.0 11 0x200508 60 absolute > 300 '1 minute load
# average' ''
# monitor oid loadAverage5Min.0 12 0x200508 300 absolute > 200 '5 minute load
# average' ''
# monitor oid loadAverage15Min.0 13 0x200508 900 absolute > 200 '15 minute load
# average' ''
# monitor oid swapCapacity.0 25 0x200508 60 absolute >= 90 'Warning: Swap
# utilization > 90%' ''
# monitor oid memCapacity.0 26 0x200508 60 absolute >= 85 'Warning: Memory
# utilization > 85%' ''
# monitor oid ntRunQLen.0 27 0x200508 60 absolute >= 3 'Warning: Run Queue
# Length >= 3' ''
#
```

```

# You can add file system monitoring entries to have the agent automatically
# monitor a file system based on the mount point. This creates an entry
# in the Empire Monitor Table.
#
# Usage:
# monitor filesystem 'name' attribute index flags interval stype oper value
# 'descr' 'action'
#
# Examples:
# monitor filesystem 'C:' devCapacity 60 0x200508 60 absolute >= 90 'C: Drive
# is > 90%' ''
# monitor filesystem 'D:' devCapacity 61 0x200508 300 absolute >= 95 'D: Drive
# is > 95%' ''
# monitor filesystem 'E:' devCapacity 62 0x200508 300 absolute >= 95 'E: Drive
# is > 95%' ''
# monitor filesystem 'F:' devCapacity 63 0x200508 300 absolute >= 95 'F: Drive
# is > 95%' ''
# monitor filesystem 'C:' devCapacity 64 0x200508 300 absolute >= 98 'C: > 98%'
# "C:\sysedge\bin\email.exe -r mymailservername sysedge@mydomain.com
# mis@mydomain.com 'C: > 98 machineX' 'C: > 98 machineX'"
#
# You can add history collection entries to the agent via
# the configuration file keyword emphistory.
#
# Usage:
# emphistory Index Interval ObjectID buckets 'Description'
#
# Examples:
# emphistory 12 120 swapCapacity.0 30 'Swap capacity history'
# emphistory 13 60 memInUseCapacity.0 60 'MemInUse capacity history'
# emphistory 14 60 memInUse.0 60 'MemInUse history'
# emphistory 11 30 loadAverage1Min.0 60 '1-min load average history'
# emphistory 15 60 numPageFaults.0 200 'Page Faults'
#
# Optional plugins
# Example for Service Response
# sysedge_plugin c:\sysedge\plugins\svcrsp\svcrsp.dll
#
# Optional plugins
# Example for Top Processes
# sysedge_plugin C:\sysedge\plugins\topprocs\topprocs.dll

```

Configuring System Information

The SystemEDGE installation enabled you to define system location and contact. If you want to modify those values, you can do so manually in the `sysedge.cf` file.

You can update the `syscontact` and `syslocation` fields as follows:

- Replace *System contact unknown* with the name of the person who is the contact for this system.
- Replace *System location unknown* with a short description of the system's physical location. For example, specify **QA Lab**.

Configuring Access Communities

The SystemEDGE installation enabled you to define read-only and read-write communities. You can modify those communities or define additional communities manually in the `sysedge.cf` file. The configuration file defines access communities using the following format:

```
community community-name permissions access-list
```

The variables are defined as follows:

You can use any ASCII characters for the community name.

- *community-name* is any octet string.
- *permissions* is either read-only or read-write.
- *access-list* is a space-separated list of IP addresses (in dotted decimal notation) that defines the systems that have access using the given community string. Access lists are not totally secure because systems can still spoof IP addresses. Access lists do, however, provide the ability to restrict legitimate use.

In the following example, SystemEDGE allows read-write access using the community-string private only to systems with one of the following IP addresses: 127.0.0.1, 45.0.4.10, 45.0.8.12, or 198.130.5.7. SystemEDGE treats any other system that attempts to use private as an authentication failure:

```
community private read-write 127.0.0.1 45.0.4.10 45.0.8.12 198.130.5.7
```

NOTE

The community string of private is used here only as an example. Do not use this value for a read-write community string. Instead, use something like eLtHakSoR97.

Use the examples in this chapter as guidelines for editing the sysedge.cf file to define your own access communities.

Specifying the Access List

If the access list is empty, SystemEDGE grants access to *any* system that uses this community string. The following restrictions apply to the access list:

- IP addresses must be separated by a space character; you cannot use any other characters, including the newline.
- The maximum length of a community string statement (including any access list) is 1024 characters, which provides enough space for about 60 IP addresses. To configure longer access lists, define separate communities.
- The SystemEDGE agent software distribution includes the edgewatch, edgemon, and emphistory command-line utilities, which act as manager systems, sending requests to the agent. If you are using any of these utilities on the same system on which the agent is installed, include that system's IP address in the access list.

Default Settings

When the SystemEDGE agent is installed, `sysedge.cf` defines a single access community named `public`, which provides read-only access to MIB objects. The definition appears as follows:

```
community public read-only
```

NOTE

Common practice allows read-only access using the community name `public`.

To modify the values of MIB objects (through SNMP Set operations), you must define a community that has read-write access permissions. For example, you could add a definition like the following to the `sysedge.cf` file:

```
community private read-write
```

Configuring Trap Communities

The `sysedge.cf` file contains definitions for trap communities, which tell the SystemEDGE agent where to send trap messages. You can configure the agent to send traps to any number of management systems. The SystemEDGE installation enables you to define trap communities. You can define additional communities as described in this section.

For each management system to which you want to send traps, add a line with one of the following formats:

- `trap_community community-name ip-address`
- `trap_community community-name hostname`

For example, add the following lines to send traps with a *community-name* of mycommunity to two systems, one with the IP address 10.16.5.26 and the other with the hostname atlanta-noc:

```
trap_community mycommunity 10.16.5.26
trap_community mycommunity atlanta-noc
```

Specifying a Trap Source

Optionally, you can specify a trap source. This parameter enables you to specify the IP address that is sent in SystemEDGE Trap protocol description units (PDUs). By default, SystemEDGE uses whatever value is returned for the system by the gethostbyname function call. Specify *trap_source* if you want to override the default behavior.

The agent has only one trap source parameter, so you set this value only once in *sysedge.cf*, and then *all* traps in all communities will take the address you have specified. The address you specify must be a valid IP address, but the SystemEDGE agent does not perform error checking to determine whether you have specified an address that is valid for the specific system you are using as the trap source.

To specify a trap source, add a line in one of the following formats to the *sysedge.cf* file to specify the source of the trap:

- *trap_source ip-address*
- *trap_source hostname*

For example, to set the trap source to a system with an IP address of 10.0.7.73 and a hostname of system1.empire.com, you can do either of the following.

To use the IP address, add the following line to *sysedge.cf*:

```
trap_source 10.0.7.73
```

To use the hostname, add the following line to `sysedge.cf`:

```
trap_source system1.empire.com
```

Configuring Authentication Failure Traps

You can configure the SystemEDGE agent to send an Authentication Failure trap whenever it receives an SNMP message whose community name does not match one of the communities recognized by the agent.

By default, the agent does *not* send Authentication Failure traps (`no_authen_traps`). To configure the agent to send Authentication Failure traps, comment out the `no_authen_traps` directive in `sysedge.cf` by placing a pound sign (#) character at the beginning of the line as follows:

```
# no_authen_traps
```

Configuring Support for Who Table Information

By default, the SystemEDGE agent supports the Who Table, which provides information about users who are currently logged into a system. For more information about this table, refer to Chapter 7, “Systems Management MIB.” Because the disclosure of this type of information could pose a potential security risk, you might want to disable the agent’s support for this information.

To disable support for the Who Table, uncomment the following line in `sysedge.cf` by removing the pound sign (#) character from the beginning of the following line:

```
# no_who_table
```

Configuring Support for User and Group Information

By default, the SystemEDGE agent supports the User table and the Group table, which provide information about the user accounts and user groups that have been configured for the system. The type of information in these tables is similar to the information that exists in the `/etc/passwd` and `/etc/group` directories. For more information, refer to Chapter 7, “Systems Management MIB.”

You might want to disable support for User and Group information in the following cases:

- Your organization considers the disclosure of user and group information to be a potential security issue.
- You have a distributed system with large numbers of users and/or groups. Because the agent periodically caches this information internally, storing user and group information could consume a significant amount of resources.

To disable support for the User and Group tables, uncomment the following line in `sysedge.cf` by removing the pound sign (#) character from the following line:

```
# no_usergroup_table
```

Configuring Support for Remote Shell Capability

By default, the SystemEDGE agent supports the Remote Shell group, which allows management systems to remotely instruct the agent to execute shell scripts and programs on the system on which the agent is running. For more information, refer to Chapter 7, “Systems Management MIB.”

Because the disclosure of this type of information could pose a potential security risk, you might want to disable the agent’s support for this information. To disable support for the Remote Shell Group, uncomment the following line in `sysedge.cf` by removing the pound sign (#) character from the following line:

```
# no_remoteshell_group
```

Configuring Alternative Syslog Facilities (UNIX Only)

The SystemEDGE agent, by default, logs all syslog messages to the LOG_DAEMON syslog facility. You can specify a different facility in the configuration file.

NOTE

Windows does not support the syslog facility. On Windows systems, all syslog output is logged by default to the %SystemRoot%\system32\sysedge.log file. If you are using Windows, refer to the next section, “Configuring Alternative Syslog Facilities (Windows Only).”

The following example sends the SystemEDGE agent’s syslog messages to the local1 facility. Add this entry near the top of the sysedge.cf file to redirect any syslog messages that are generated by errors in sysedge.cf:

```
syslog_facility local1
```

Configuring Alternative Syslog Facilities (Windows Only)

On Windows systems, the SystemEDGE agent logs all syslog messages to the %SystemRoot%\system32\sysedge.log file by default. That file size is unlimited. However, you can use the syslog_logfile directive to specify an alternative sysedge.log location (and file name), and to place limits on the size of that log file and the number of old log files that the agent saves.

Use the syslog_logfile directive as follows:

```
syslog_logfile filename size number
```

The variables are defined as follows:

- *filename* is the complete path to the desired log file.
- *size* is the maximum file size in KB.
- *number* is the number of log files to preserve for historical purposes. A minimum of two files is recommended.

For example, to instruct the SystemEDGE agent to log messages to the file `C:\sysedge\sysedge.log`, creating up to two log files with a maximum size of 20 KB each, add the following to the `sysedge.cf` file:

```
syslog_logfile c:\sysedge\sysedge.log 20 2
```

Configuring Support for Agent Debugging

By default, the SystemEDGE agent logs all syslog messages of priority `LOG_INFO` or lower. (Lower priority levels signify greater importance.) While UNIX-based agents can change this log-level through the command-line `-d` option, the SystemEDGE agent on Windows cannot, because no command-line options are available to it. Consequently, to use the configuration file option to instruct the SystemEDGE agent (on both UNIX and Windows systems) to log messages of priority `LOG_DEBUG` or lower, uncomment the following line by removing the pound sign (`#`) character:

```
# sysedge_debug
```

For more information on the syslog facility, refer to Appendix B.

NOTE

On Windows systems, all syslog output is logged to the file `%SystemRoot%\system32\sysedge.log`. For more information, refer to “Configuring Alternative Syslog Facilities (Windows Only)” on page 134.

Configuring Support for Floppy Status Checking

By default, the SystemEDGE agent automatically determines the status of all floppy devices on the system as part of its support for the `hrDeviceTable` from the Host Resources MIB. On some UNIX systems, however, when you check the status (with the `stat` command) of a floppy device that contains no media, the console and some system log files display warning

messages. To circumvent these warning messages, configure the agent to not check status of floppy drives. To do so, uncomment the following line in `sysedge.cf` by removing the pound sign (#) character:

```
# no_stat_floppy
```

Configuring Support for Serial Port Status Checking

By default, the SystemEDGE agent automatically determines the status of all serial devices on the host system as part of its support for the `hrDeviceTable` from the Host Resources MIB.

Some serial applications, however, encounter problems because they are unable to handle the opening and closing of serial devices (which are necessary for determining status) by any process other than themselves. For example, some `tty` and serial applications become confused when another application briefly opens and closes a serial port device.

To circumvent these problematic serial applications, use the `no_serial_status` configuration option to configure the agent to check only the keyboard and mouse. In this case, the agent returns `unknown(1)` for the status of serial ports when it is queried by management systems.

To inhibit serial port status checking, uncomment the following line in `sysedge.cf` by removing the pound sign (#) character from the following line:

```
# no_serial_status
```

Configuring Support for Disk Probing

By default, the SystemEDGE agent automatically determines size, capacity, description, and other properties of disks and CD-ROMs that may be installed on the underlying system. The agent usually uses I/O control functions (for example, UNIX

ioctl) to obtain this information. However, on some older UNIX systems (for example, HP-UX), probing of disk devices may cause the agent to block while the driver waits on status information.

You can use the `no_probe_disks` option to avoid potentially lengthy agent blocking. To inhibit disk probing, add the following line to `sysedge.cf`:

```
no_probe_disks
```

NOTE

If this option is enabled, the agent may be unable to provide disk statistics, capacity information, device descriptions, and status information.

Configuring Support for Actions

By default, the SystemEDGE agent allows the execution of action commands in conjunction with the self-monitoring tables. The capability to execute action commands and scripts could be a potential security issue because the command and scripts could execute commands as the root or administrator users. Depending on the local security policies in effect at your site, you may wish to disable the agent's support for executing action commands. For more information about actions, refer to the following sections:

- “Monitor Table Actions” on page 255
- “Process Monitor Table Actions” on page 296
- “Process Group Monitor Table Actions” on page 326
- “Log Monitor Table Actions” on page 340
- “NT Event Monitor Table Actions” on page 366

To disable support for action execution, add the following line to `sysedge.cf`:

```
no_actions
```

Disabling Support for Remote File System Checking (UNIX Only)

By default, the SystemEDGE agent makes data about all file systems (local and remote) available through the Systems Management MIB. However, on some UNIX systems, SystemEDGE can be blocked if a file system is mounted from a remote file server that is no longer available (either if it is down or if the network connection between the server and client is down). Unfortunately, there is no way for the SystemEDGE agent to unblock in this situation. To circumvent this blocking, configure SystemEDGE agent to avoid checking status on remote file systems. To do so, add the following line to the `sysedge.cf` file:

```
no_stat_nfs_filesystems
```

NOTE

If you add this line to the `sysedge.cf` file, it disables the checking of *all* remote file systems—not just NFS file systems.

Configuring Support for Threshold Monitoring

The SystemEDGE agent includes support for threshold monitoring of MIB objects, including file systems, interfaces, processors, and so on. You can use SNMP Set requests to add entries for threshold monitoring dynamically while the agent is running, or you can define them in the `sysedge.cf` configuration file that the SystemEDGE agent reads when it starts. For more information about creating entries in the Monitor table, refer to Chapter 10, “Configuring Threshold Monitoring.”

Configuring Support for Process Monitoring

By default, the SystemEDGE agent allows SNMP Gets and Sets to the Systems Management Process Monitor table and the Host Resources Running Software table, assuming that queries use a valid community with read-only or read-write permissions (respectively).

Performing SNMP Sets in those tables could be a potential security issue: SNMP Sets can send UNIX processes signals (for example, KILL) and can terminate processes on Windows systems. Depending on the local security policies that are in effect at your site, you may want to disable the agent's support for SNMP Sets in these tables. For more information about these tables, refer to Chapter 7, “Systems Management MIB” and Chapter 9, “Host Resources MIB.”

To disable support for SNMP Sets to the Process Monitor table, add the following line to `sysedge.cf`:

```
no_process_sets
```

Performing SNMP Gets in those tables could also be a potential security issue: SNMP Gets can discover the processes that are running on the underlying system. Depending on the local security policies that are in effect at your site, you may want to disable the agent's support for SNMP Gets in these tables. For more information about these tables, refer to Chapter 7, “Systems Management MIB” and Chapter 9, “Host Resources MIB.”

To disable support for SNMP Gets and Sets to the Process Monitor table, add the following line to `sysedge.cf`:

```
no_process_table
```

Monitoring Applications, Processes, and Services

The SystemEDGE agent can also monitor applications, processes, and Windows services by creating entries in the Process Monitor table. You can dynamically add entries through an SNMP Set request while the agent is running, or you can define them in the `sysedge.cf` configuration file that the agent reads when it starts.

Monitoring Process Attributes

The `watch process` configuration file directive automatically configures the agent to monitor a process attribute that you specify. You identify the process to be monitored using regular expressions to match the process name and the attribute of the process that you want to monitor. The SystemEDGE agent automatically determines the process ID for the specified process and then creates the appropriate entry in the agent's Process Monitor table. When you use the `watch process` directive, you do not need to know the process ID or to use SNMP Set requests to add an entry to the Process Monitor table. For more information about creating entries in the Process Monitor table, refer to Chapter 11, "Configuring Process and Service Monitoring."

Monitoring Windows Services

The `watch ntservice` configuration file directive automatically configures the agent to monitor a Windows service to ensure that it is running. You identify the Windows service that you want to monitor, and the SystemEDGE agent automatically determines the service index from the NT Service MIB table and creates the appropriate entry in the agent's Process Monitor table. For more information about the NT Service MIB table, refer to Chapter 7, "Systems Management MIB."

Configuring Support for Process Group Monitoring

The flexible Process Group Monitor table of the Systems Management MIB enables you to dynamically configure the SystemEDGE agent to monitor groups of processes that are running on the underlying system. You select the process group, regular expression, and interval, and the agent uses that information to monitor those process groups. For example, the agent can determine what processes exist in each group and whether the group membership changes. If components of an application start or fail, or if members leave a group or are added to a group, the SystemEDGE agent can automatically notify the NMS. For more information about creating entries in the Process Monitor table, refer to Chapter 12, “Configuring Process Group Monitoring.”

Configuring Support for Log File Monitoring

The SystemEDGE agent includes a log file monitoring capability that lets you instruct the agent to monitor log files continuously for the appearance of user-specified regular expressions, and to notify the management system with a trap message if the agent finds a match. You can specify entries for log file monitoring dynamically (through SNMP Set requests) while the agent is running, or you can define them in the `sysedge.cf` configuration file. For more information about creating entries in the Process Monitor table, refer to Chapter 13, “Configuring Log File Monitoring.”

Configuring Support for Windows Event Log Monitoring (Windows Only)

The SystemEDGE agent includes a Windows Event Monitoring capability that lets you instruct the agent to continuously monitor Windows event logs in much the same way that it monitors textual log files. Whenever a matching event is generated on the system, the agent notifies the management system with a trap message and can execute an action command to immediately handle the event. Because Windows events

include several identifying characteristics in addition to the textual message, this monitoring capability enables you to specify more sophisticated types of matches. For more information about creating entries in the Process Monitor table, refer to Chapter 14, “Configuring Windows Event Monitoring.”

Configuring History Collection

The SystemEDGE agent can track the value of various integer-based MIB objects (counters, gauges, and so on) over time and can store them for later retrieval. You can define entries for history collection in the `sysedge.cf` configuration file that the SystemEDGE agent reads when it starts. For more information about creating entries in the Process Monitor table, refer to Chapter 15, “Configuring History Collection.”

Configuring User/Group Permissions for Subprograms (UNIX Only)

By default, the SystemEDGE agent runs subprograms (for example, remote shell, action, and extension object invocations) with its effective user and group permissions—normally root. Depending on the local security policies in effect at your site, you may want to set the agent to use actions and extension objects that run with different user and group permissions.

To run subprograms with the effective user and group permissions of a user other than root, add the following statements to your SystemEDGE agent configuration file:

```
subprogram_user_name concord  
subprogram_group_name sysmgmt
```

In these examples, all subprograms execute with the effective permissions of the user `concord` and group `sysmgmt`. The user and group names that you specify *must be valid* on the underlying system.

NOTE

If either the user name or group name is incorrect, SystemEDGE disables all subprogram functionality. That is, the agent does not support actions, extension MIB objects, and remote-shell capabilities.

Configuring the SNMP Bind Address

By default, SystemEDGE binds to all interfaces (* /UDP-161). You can change the port to which SystemEDGE binds by using the -p option when you invoke SystemEDGE, or you can bind SystemEDGE to a specific interface by using the bind_address token as follows:

```
bind_address ip-address
```

For example, to bind to the 10.1.0.202 address only, enter the following in sysedge.cf:

```
bind_address 10.1.0.202
```

Configuring Support for eHealth AIMs

The SystemEDGE agent provides a plug-in architecture through which you can load optional eHealth AIMs at initialization. These eHealth AIMs provide an extensible and flexible approach to supporting application-specific MIB variables. For the most current list of AIMs, refer to the Product Quick List page of the eHealth product Web site (<http://www.concord.com/products/quicklist.shtml>).

By default, the SystemEDGE agent does not load any AIMs at initialization time. You can edit the sysedge.cf file to specify which AIMs the agent should load. You must specify absolute paths to enable the SystemEDGE agent to find the AIM to load.

NOTE

The Top Processes AIM is included with the SystemEDGE distribution on every platform. This AIM does not require an additional license to operate. You can set the agent to load the Top Processes AIM during the SystemEDGE agent installation.

To load the AIM for Top Processes from the standard Solaris distribution directory, for example, enter the following in the `sysedge.cf` file:

```
sysedge_plugin /opt/EMPSysedge/plugins/topprocs.so
```

To load the AIM for Top Processes module from the standard Windows distribution directory, enter the following in the `sysedge.cf` file:

```
sysedge_plugin c:\sysedge\plugins\topprocs.dll
```

NOTE

If you selected the option for configuring Top Processes during the SystemEDGE installation, this line was automatically added to the `sysedge.cf` file.

For information about enabling other AIMs, refer to the user guide for that AIM.

Configuring Support for Linux Free Memory

Linux free memory is calculated as total physical memory less memory in use. By default, memory in use includes system buffers and disk cache.

Cache and system buffers can be reclaimed by the operating system if memory is needed for processes. For this reason, some choose to view free memory as including memory that is used by the operating system for caching or system buffers. The `linux_freemem_include` directive is used to include these values in the Linux free memory calculation. The directive supports the following two options:

- **buffers** – include system buffers in the free memory calculation.
- **cached** – include cached memory in the free memory calculation.

At least one of the above options must be specified. The options may be specified in any order and must be separated by one or more spaces.

For example, to include both buffers and cached memory in free memory add the following line to `sysedge.cf`:

```
linux_freemem_include buffers cached
```

Recommendations for Configuring Security

Table 11 lists recommended configuration options for ensuring security.

Table 11. Recommended Settings for Ensuring Secure Systems

Command	Refer to this Section
no_who_table	“Configuring Support for Who Table Information” on page 132
no_usergroup_table	“Configuring Support for User and Group Information” on page 133
no_remoteshell_group	“Configuring Support for Remote Shell Capability” on page 133
no_process_sets	“Configuring Support for Process Monitoring” on page 139

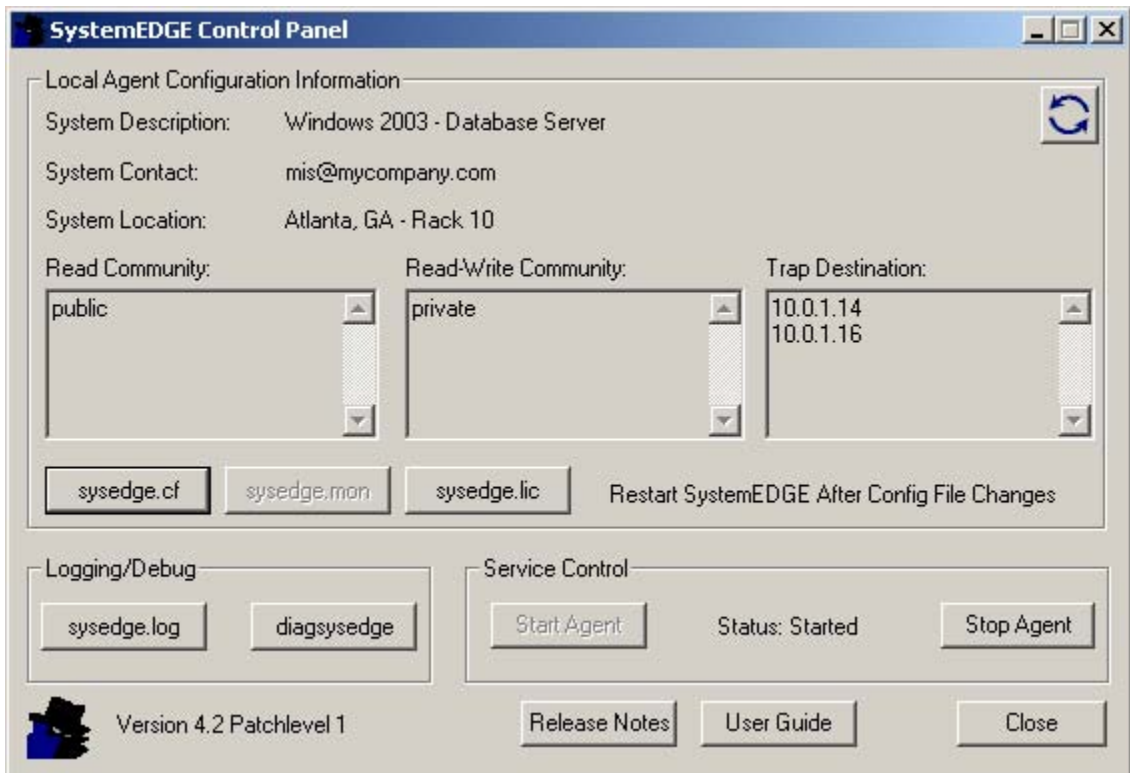
Table 12 lists additional configuration options that can help ensure security.

Table 12. Additional Settings for Ensuring Secure Systems

Command	Refer to this Section
no_actions	“Configuring Support for Actions” on page 137
no_process_table	“Configuring Support for Process Monitoring” on page 139
subprogram_user_name	“Configuring User/Group Permissions for Subprograms (UNIX Only)” on page 142
subprogram_group_name	“Configuring User/Group Permissions for Subprograms (UNIX Only)” on page 142

Using the SystemEDGE Control Panel for Windows

As a standalone Windows service, SystemEDGE 4.2 and later has its own SystemEDGE Control Panel. To view the SystemEDGE Control Panel, open the Control Panel dialog box and double-click **eHealth SystemEDGE**.



You can use the SystemEDGE Control Panel to do any of the following:

- Start and stop the agent.
- View community strings and trap destinations.
- Open the configuration and license files.
- View the SystemEDGE log file.
- Run the diagsysedge utility.
- View the *eHealth SystemEDGE User Guide* and the *eHealth SystemEDGE Release Notes*.

Starting the SystemEDGE Agent

This chapter explains how to start the SystemEDGE agent. Before you do, you must license the agent and configure it for your environment. For more information, refer to Chapter 3, “Licensing the SystemEDGE Agent” and Chapter 4, “Configuring the SystemEDGE Agent.”

To start the agent manually, refer to the next section, “Starting the Agent Manually.” To start the agent at system boot, refer to “Starting the Agent Automatically at System Boot” on page 151.

NOTE

After you start the agent as described in this chapter, you can use the `diagsysedge.exe` program to verify that your agent is running. For more information, refer to “Using `diagsysedge.exe`” on page 423.

Starting the Agent Manually

You can start the agent manually for both Windows and UNIX systems.

Starting SystemEDGE on Windows Systems

For Windows systems, you can manually start SystemEDGE from the command line, the Services Control Panel, or the SystemEDGE Control Panel.

To start SystemEDGE from the command line:

1. Enter the following:

```
net start sysedge
```

To start SystemEDGE from the Services Control Panel:

1. Select **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Services**.
2. Right-click **SystemEDGE**, and select **Start**.

To start SystemEDGE from the SystemEDGE Control Panel:

1. Select **Start** → **Settings** → **Control Panel**.
2. Double-click **eHealth SystemEDGE**, and click **Start Agent**.

NOTE

In the Windows XP Category View, the SystemEDGE control panel is under "Network and Internet Connections." In the Windows XP Classic View, the SystemEDGE control panel icon is at the main level of the display.

Starting SystemEDGE on UNIX Systems

To start the SystemEDGE agent from the UNIX command line, log in as root and then enter the following:

```
bin/sysedge -b
```

Command Line Options for UNIX Systems

This section describes the command line options for starting the SystemEDGE agent. The usage options are as follows:

```
sysedge [-b] [-d] [-f configfile] [-l licensefile] [-p port] [-h] [-t]
```

Table 13 describes the UNIX command-line options for starting the SystemEDGE agent.

Table 13. UNIX Command-Line Options for Starting the SystemEDGE Agent

Option	Description
-b	Run the agent in the background. The SystemEDGE agent runs as a daemon process and disconnects from the controlling terminal. Use this flag when starting the SystemEDGE agent from a startup script.
-d	Run the agent in debug mode. This option causes the agent to log debug-level messages with the syslog facility. For more information about syslog, refer to Appendix B.
-f <i>configfile</i>	Read configuration parameters from the <i>configfile</i> file instead of the default <i>/etc/sysedge.cf</i> file.
-h	Display help for the command. This option lists the available command line options.
-l <i>licensefile</i>	Use the license key <i>licensefile</i> file instead of the default <i>/etc/sysedge.lic</i> file.
-m <i>monitorfile</i>	Use the Monitor table configuration file <i>monitorfile</i> instead of the default <i>/etc/sysedge.mon</i> file.
-p <i>port</i>	Listen for incoming SNMP messages on <i>port</i> instead of on the standard SNMP port 161. CA has reserved UDP/1691 for use as an alternate port for running the SystemEDGE agent.
-t	Run the agent in packet-trace mode. This option causes the agent to write a packet dump of each SNMP PDU received or transmitted by the agent to standard output (file descriptor 1).

Starting the Agent Automatically at System Boot

You can configure your system to automatically start the SystemEDGE agent whenever the system is booted.

NOTE

If you answered yes to the installation script prompt, “Disable native SNMP Agent if applicable?” the installation script configured your system to start the SystemEDGE agent automatically whenever the system is booted.

If you did not configure your system to start the SystemEDGE agent at system boot during the installation process, you can do so by following the instructions in this section. This section includes instructions for the following operating systems:

- Solaris
- Windows (page 153)
- HP-UX (page 153)
- Linux (page 154)
- AIX (page 154)
- Digital UNIX or Tru64 (page 155)

Starting the Agent Automatically for Solaris Systems

For Solaris 2.x systems, the SystemEDGE agent’s config subdirectory contains a script named `S99sysedge`. The installation process places this script in `/etc/rc2.d` to start the agent automatically at boot time.

If you installed the agent in a directory other than the default (`/opt/EMPsysedge`), you must edit the `S99sysedge` script to reflect those changes. In addition, you must edit the script to include the correct directory for the agent’s configuration files (`sysedge.lic`, `sysedge.cf`, and `sysedge.mon`) if you installed them in a directory other than the default (`/etc`).

Starting the Agent Automatically for Windows Systems

The SystemEDGE service starts by default at system boot time. To verify that the agent is set to start automatically, open the Services Control Panel, right-click **SystemEDGE**, and select **Properties**. On the General tab, select **Automatic** from the **Startup type** list, and then click **OK**.

Starting the Agent Automatically for HP-UX Systems

For HP-UX version systems, you must run the SystemEDGE agent's sysedge script to start the agent automatically at boot time. Copy this script from the config subdirectory to /sbin/init.d and then create a symbolic link to the file by entering the following command:

```
ln -s /sbin/init.d/sysedge /sbin/rc2.d/S990sysedge
```

If you installed the agent in a directory other than the default (/opt/EMPsysedge), you must edit sysedge to include the correct installation directory. In addition, you must edit sysedge to include the correct directory for the agent's three configuration files (sysedge.lic, sysedge.cf, and sysedge.mon) if you installed them in a directory other than the default (/etc).

To run the SystemEDGE agent instead of the native HP-UX agent, edit the following statements in /etc/rc.config.d, as described:

- 'SNMP_MASTER_START=1' to 'SNMP_MASTER_START=0'
- 'SNMP_HPUNIX_START=1' to 'SNMP_HPUNIX_START=0'
- 'SNMP_MIB2_START=1' to 'SNMP_MIB2_START=0'

Starting the Agent Automatically for Linux Systems

For Linux systems, you must run the SystemEDGE agent's `sysedge` script to start the agent automatically at system boot time. Copy this script from the `config` subdirectory to `/etc/init.d` and then create a symbolic link to the file by entering the following command:

```
ln -s /etc/rc.d/init.d/sysedge /etc/rc.d/rc2.d/S99sysedge
```

If you installed the agent in a directory other than the default (`/opt/EMPsysedge`), you must edit the `sysedge` script to include the correct installation directory. If you installed the agent's three configuration files (`sysedge.lic`, `sysedge.cf`, and `sysedge.mon`) in a directory other than the default (`/etc`), you must also edit `sysedge` to include the correct directory for these files.

NOTE

If you are not using the directory structure that the Red Hat Linux distribution uses, you must edit the commands for starting the agent to match the directory structure for the distribution you are using.

To run the SystemEDGE agent instead of the native Linux agent, edit `/etc/rc.d/init.d/snmpd` by entering an `exit 0` command before any of the `snmpd` script's code.

Starting the Agent Automatically for AIX Systems

To start the agent automatically for AIX systems, you must add the commands for starting the agent to the local boot file `/etc/rc.tcpip`. To do so and to include the `sysedge` agent process, enter the following:

```
/usr/lpp/EMPsysedge/bin/sysedge -b
```

When you enter this command, you must include the full directory path and file name for sysedge on your system.

Starting the Agent Automatically for Digital UNIX or Tru64 Systems

For Digital UNIX and Tru64 systems, you must run the SystemEDGE agent's sysedge script to start the agent automatically at system boot time. Copy this script from the config subdirectory to /etc/init.d and then create a symbolic link to the file by entering the following command:

```
ln -s /sbin/init.d/sysedge /sbin/rc3.d/S99sysedge
```

If you installed the agent in a directory other than the default (/usr/opt/EMPsysedge), you must edit the sysedge script to include the correct installation directory. If you installed the agent's three configuration files (sysedge.lic, sysedge.cf, and sysedge.mon) in a directory other than the default (/etc), you must also edit sysedge to include the correct directory for these files.

To run the SystemEDGE agent instead of the native Digital UNIX or Tru64 agent, edit /sbin/init.d/snmpd by entering an exit 0 command before any of the snmpd script's code.

Logging Agent Operation Messages

This section explains how to log agent operation messages.

Logging Messages for UNIX

The SystemEDGE agent uses the UNIX syslog facility to log informational messages and error conditions that it may encounter during operation. The syslog daemon typically logs these messages to the /var/adm/messages text file, depending on how the syslog daemon is configured on your system. By

default, the agent daemon uses syslog to log messages of priority levels informational through emergency. If you are running the agent in debug mode through the -d runtime command line option, the agent also logs messages of priority level debug.

For information about configuring the syslog daemon on your system to log messages from daemon processes like sysedge to the /var/adm/daemon-log text file, refer to Appendix B. For more information about syslog, refer to the following Man pages: syslog(3), syslog.conf(5), and syslogd(8).

Logging Messages for Windows

The SystemEDGE agent logs informational messages and error conditions that it may encounter during operation in the %SystemRoot%\system32\sysedge.log. For information about possible problems that may arise during SystemEDGE agent operation, refer to the sysedge.log file.

Using the SystemEDGE Agent with Other SNMP Agents

This chapter describes how to use the SystemEDGE agent in conjunction with other SNMP agents.

Supporting Multiple SNMP Agents

By default, all SNMP agents attempt to use UDP port 161, but only a single process can bind to a given port at any one time. That is, only one SNMP agent can use UDP port 161 at a time. If you are running multiple SNMP agents simultaneously, you must develop strategies for their coexistence.

The following are potential solutions for enabling several SNMP agents to share UDP port 161:

- Multiplex UDP port 161 among several SNMP agents, usually in a master/slave relationship.
- Execute agents simultaneously if each binds to a separate UDP port and management software is configured to communicate with them individually.
- Code agents to conform to every existing proprietary master/subagent API. This solution has an exceedingly high cost in terms of coding, testing, and licensing the numerous proprietary APIs.

UDP port 1691 is reserved for use by the SystemEDGE agent. You can configure the agent to use that port instead of UDP/161. For more information about starting the SystemEDGE agent on an alternate port, refer to Chapter 5, "Starting the SystemEDGE Agent."

The most common notion of agent multiplexing involves a master agent that is bound to UDP port 161 communicating with separate subagents or slave agents that are implementing different MIBs. The master and subagents communicate using a defined protocol for registering the subagent and exchanging messages between them. Several protocols exist for governing such a communication protocol between Master and subagent.

Agent Multiplexing

The two most common multiplexing protocols are SNMP-Distributed Protocol Interface (DPI) and SNMP multiplexing (SMUX). **SNMP-DPI** is an extension to SNMP agents that permits users to dynamically add, delete, or replace management variables in the local MIB through a subagent without having to recompile the SNMP agent. **SMUX** is a session-management protocol that provides a lightweight communication channel from the underlying transport layer to the application layer by multiplexing data streams on top of a reliable stream-oriented transport.

*For more information about SNMP-DPI, refer to RFC 1592.
For more information about SMUX, refer to RFC 1227.*

Neither protocol is standard or dominant in the marketplace. The IETF is currently working on a standardized approach based on SNMP-DPI called AgentX. These RFCs define only the protocol that is used between Master and subagent. They do *not* define a set of APIs. Because the specifications for DPI and SMUX are publicly available, an agent developer can implement a master or subagent that conforms to either specification without encountering licensing and royalty fees when using them with another vendor's master or subagent. However, because neither protocol is standard, an agent developer has to support both protocols to ensure the greatest amount of master-slave agent multiplexing interoperability.

Monolithic Agents

Several proprietary, non-standard, agent-multiplexing solutions exist, although they are not technically true agent multiplexing. These solutions are based on the notion of monolithic agents, whereby subagents are actually linked into the Master agent's executable binary or its address space while they are running. This task can be accomplished without Master agent source code, as long as the subagents adhere to the proprietary API.

There are several drawbacks to this approach:

- The agent developer may have to support multiple code bases because the proprietary APIs are often incompatible with those used within other agent source bases.
- Because the API is proprietary, these subagents cannot communicate with other, more open specifications, such as DPI and SMUX.
- The development and deployment of subagents usually involve licensing and royalty fees, which are usually charged on a per-CPU basis. That means that the more subagents you deploy, the more the user must pay in licensing and royalty fees. In addition, because this solution is proprietary, every developer of the Master agent and subagents must purchase those APIs to make the pieces interoperate.

Using the SystemEDGE Agent with the Solstice Enterprise Agent

When you run the SystemEDGE agent on UDP port 1691, you can use it as a subagent under the Solstice Enterprise Agent (SEA). The SystemEDGE agent installation package installs the appropriate configuration file. Table 14 describes the SEA configuration files.

Table 14. SystemEDGE Configuration Files for SEA

File Name	Description
<code>sysedge.reg</code>	Registration file that specifies the MIB branches to query the SystemEDGE agent and the UDP port number to which to send SNMP queries.
<code>sysedge.rsrc</code>	Resource configuration file that specifies the SystemEDGE agent type and the location of its corresponding registration file (by default, <code>/etc/snmp/conf/sysedge.reg</code>).

To configure the SystemEDGE agent to run as a subagent under the SEA:

1. Install the SEA packages that correspond to your version of Solaris and the underlying hardware architecture.
2. Copy the SystemEDGE agent SEA configuration file into the SEA configuration directory, `/etc/snmp`, by entering the following commands from the directory where you installed the SystemEDGE agent:

```
cp config/sysedge.reg /etc/snmp/conf  
cp config/sysedge.rsrc /etc/snmp/conf
```


3. Restart the SystemEDGE agent so that it will use port UDP/1691 instead of the default UDP/161.

The S99sysedge script attempts to determine whether to use the SystemEDGE agent in conjunction with the SEA. It does so by checking for the existence of the `/etc/snmp/conf` directory and the `/etc/rc2.d/K76snmpdx` file. If the script determines that it should use both agents, it instructs the SystemEDGE agent to use port UDP/1691.

4. Restart the SEA multiplexor by entering the following commands:

```
/etc/rc2.d/K76snmpdx stop  
/etc/rc2.d/K76snmpdx start
```

NOTE

Although the SEA multiplexes SNMP Get, GetNext, and Set operations, it does not correctly support SNMP row-creation operations across subagents because they occur as side effects of Set operations on non-existent rows. Consequently, when you are using utilities such as `edgwatch`, `edgemon`, and `emphistory` to create rows in their respective SNMP tables, query the SystemEDGE agent directly (not the SEA master agent). You can instruct those utilities to query UDP port 1691 by appending `:1691` to the hostname or IP address when you invoke the utilities.

Using the SystemEDGE Agent with the Microsoft Windows SNMP Agent

The Microsoft Windows SNMP Agent is a master agent that multiplexes among subagents that are linked into its address space at run-time. Subagents are implemented as Windows DLLs. Upon initialization, the Microsoft master agent utilizes registry settings to determine which subagents (and corresponding DLLs) to load. Upon initialization, the subagents inform the Master agent which MIB branches they implement.

In Release 4.2, the SystemEDGE agent is no longer a subagent of the Microsoft SNMP agent. It can coexist with the Microsoft Windows extensible agent.

Take either of the following coexistence approaches:

- Run the Microsoft and SystemEDGE agents together on the system, running the SystemEDGE agent on port 1691 and the Microsoft agent on port 161.
- Turn off or disable the Microsoft Master agent, and run the SystemEDGE agent on port 161.

Using the SystemEDGE Agent with the HP SNMP Agent

HP-UX systems ship with an extensible agent and subagents that provide support for MIB-II and the HP (rather limited) private-enterprise MIB. The Master/subagent interface is based on a proprietary API that the SystemEDGE agent does not support.

Take either of the following coexistence approaches:

- Run the HP and SystemEDGE agents together on the system, running the SystemEDGE agent on port UDP/1691.
- Turn off or disable the HP master or subagent, and run the SystemEDGE agent on port UDP/161.

Using the SystemEDGE Agent with the AIX SNMP Agent

AIX Release 4.1 and later systems ship with a MIB-II SNMP agent that also supports SMUX. You can run the SystemEDGE agent in addition to or in place of the AIX agent.

Take either of the following coexistence approaches:

- Run the SystemEDGE and AIX agents together. To do so, make sure that the SystemEDGE agent is invoked with the `-p 1691` option in the AIX TCP/IP startup script, `/etc/rc.tcpip`.
- Run the SystemEDGE agent instead of the AIX agent. To do so, comment out the AIX agent's invocation in `/etc/rc.tcpip` by adding a pound sign (#) at the beginning of the following lines:

```
# Start up the Simple Network Management Protocol (SNMP) daemon
# start /usr/sbin/snmpd "$src_running"
```

Using the SystemEDGE Agent with the Digital UNIX or Tru64 SNMP Agent

Digital UNIX and Tru64 systems ship with an extensible agent and subagents that provide support for MIB-II and for the Digital and Tru64 implementation of the Host Resources MIB. The Master/subagent interface is based on a proprietary API that the SystemEDGE agent does not support.

Take either of the following coexistence approaches:

- Run the agents together on the system, running the SystemEDGE agent on port UDP/1691.
- Turn off or disable the Digital or Tru64 Master/subagent, and run the SystemEDGE agent on port UDP/161.

Using the SystemEDGE Agent with the Compaq Insight Manager

You can run the SystemEDGE agent and the Compaq Insight Manager (CIM) on the same system. When you are deploying SystemEDGE agents from AdvantEDGE View to a system that includes CIM, AdvantEDGE View stops the following CIM services:

- Compaq Foundation Agent
- Compaq Web Agent
- Compaq Storage Agent
- Compaq Server Agent
- Compaq NIC Agent

After the SystemEDGE agent and *eHealth* AIMs are deployed, AdvantEDGE View restarts all of these services.

Systems Management MIB

This chapter gives examples of management information that is available through the enterprise-specific Systems Management MIB. The Systems Management MIB module defines a collection of objects for managing host systems. The MIB is organized into sections for the following types of information:

- Host configuration
- Kernel configuration
- Mounted devices
- Users and groups
- Remote command execution
- Processes
- Streams
- Performance monitoring
- Interprocess communications (IPC)
- NFS and RPC statistics
- Buffer statistics for network buffers
- Streams buffers
- I/O buffer cache

The MIB sections are described in more detail in the following sections. The figures show sample output from a Sun SPARCstation that is running Solaris 2.5 and the SystemEDGE agent daemon, sysedge. For a complete description of the MIB

objects, refer to the `empire.asn1` file in the `/doc` subdirectory of the SystemEDGE agent distribution. Some objects cannot be implemented on all platforms. For information about platform-specific support, refer to the *eHealth SystemEDGE Release Notes*.

Host System Information

You can retrieve values of the objects in the Host System group to determine the following types of information:

- Hostname
- Operating system version and release number
- CPU type
- Amount of memory
- Version of the SystemEDGE agent

Table 15 provides sample values for each MIB object.

Table 15. Host System Information MIB Objects

MIB Object	Sample Value
nodename(1)	pluto
cpu(2)	sun4u
memory(3)	16280
hostid(4)	57212aab
osver(5)	Generic_103640-31
osrel(6)	5.8
agentVersion(8)	SystemEDGE Agent Release 4.1

Mounted Devices

For information about the devices and file systems that are mounted on the host, retrieve the Mounted Devices table. Each row in the table represents a device that is currently mounted and contains columns that represent the following types of information:

- Device's mount point
- Block size
- Total number of blocks
- Total number of free blocks
- Total number of files
- Total number of free files
- Percent capacity (percent used)

Figure 4 shows a sample Mounted Devices table.

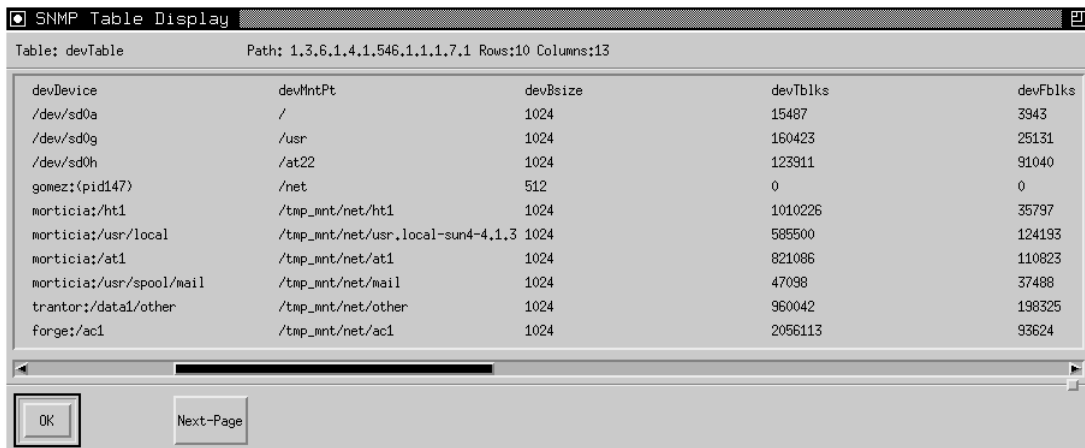


Table: devTable Path: 1.3.6.1.4.1.546.1.1.1.7.1 Rows:10 Columns:13

devDevice	devMntPt	devBsize	devTbLks	devFbLks
/dev/sd0a	/	1024	15487	3943
/dev/sd0g	/usr	1024	160423	25131
/dev/sd0h	/at22	1024	123911	91040
gomez:(pid147)	/net	512	0	0
morticia:/ht1	/tmp_mnt/net/ht1	1024	1010226	35797
morticia:/usr/local	/tmp_mnt/net/usr.local-sun4-4.1.3	1024	585500	124193
morticia:/at1	/tmp_mnt/net/at1	1024	821086	110823
morticia:/usr/spool/mail	/tmp_mnt/net/mail	1024	47098	37488
trantor:/data1/other	/tmp_mnt/net/other	1024	960042	198325
forge:/ac1	/tmp_mnt/net/ac1	1024	2056113	93624

OK Next-Page

Figure 4. Systems Management MIB: Mounted Devices Table

Monitoring File System Space

The **devCapacity** column, which shows the percentage of file system space being used, is ideal for checking for file systems that are in danger of becoming too full. You can use the Monitor table to instruct the agent to monitor the **devCapacity** values for you and to send a trap to the management system if the value goes above the threshold value that you select. For example, you can have the agent monitor the /usr directory and send a trap if it becomes greater than 90% full. For more information about setting thresholds, refer to Chapter 10, “Configuring Threshold Monitoring.”

Unmounting a Mounted Device

When you are unmounting devices, use a community name that allows you read-write access.

You can unmount a mounted device by setting the **devUnmount** column for that device to the value delete(1). When you do so, the agent unmounts the device and removes its entry from the Mounted Devices table.

Kernel Configuration

You can identify the version of the kernel that is running on the system and determine how the kernel is configured by retrieving objects in the Kernel Configuration group. Kernel configuration parameters include the following:

- Maximum number of processes that can run concurrently
- Number of CPUs
- Clock rate
- Amount of virtual memory
- Maximum number of inodes, open files, and clists
- Amount of system swap space
- Maximum memory and open files allowed per process
- Kernel version description string

Table 16 provides sample values for each MIB object in the Kernel Configuration group.

Table 16. Kernel Configuration Group MIB Objects

MIB Object	Sample Value
maxProcs(1)	138
serialNumber(2)	11931680
romVersion(3)	2.6
numCPU(4)	1
clockHZ(5)	100
kernelVers(6)	Generic_103640-31
virtualMem(7)	68596 (KB)
maxInode(8)	322
maxFiles(9)	582
maxClist(10)	228
maxMemPerProc(11)	13852 (KB)
totalSwap(12)	65516 (KB)
openMaxPerProc(13)	256
posixJobCtrl(14)	true(1)
posixVersion(15)	198808
pageSize(16)	4096

Boot Configuration

You can use the MIB objects in the Boot Configuration group to determine which device and partition the system is using for the root file system, the dump file system, and swap space, as well as the number of blocks contained by each.

Table 17 provides sample values for the MIB objects of the Boot Configuration group.

Table 17. Boot Configuration Group MIB Objects

MIB Object	Sample Value
rootName(1)	sd0a
rootFSType(2)	4.2
rootBlocks(3)	0
dumpName(4)	sd0b
dumpFSType(5)	spec
dumpBlocks(6)	131040
swapName(7)	sd0b
swapFSType(8)	spec
swapSize(9)	131040

Streams Group

The Streams I/O subsystem provides a data transit/processing path between applications in user space and drivers in kernel space in the host. You can monitor the health of the Streams subsystem by retrieving objects that provide the following types of information:

- Maximum stream message size
- Number of streams in use
- Maximum number of streams
- Number of stream allocation failures
- Number of stream queues in use
- Number of stream queue allocation failures
- Statistics for stream message blocks and data blocks

Table 18 describes the MIB objects in the Streams group and provides sample values for each.

Table 18. Streams Group MIB Objects

MIB Object	Description	Value
maxmsgSize(1)	Maximum streams message size in bytes	4096 bytes
maxNumPush(2)	Maximum number of stream modules that can be pushed at one time	9
numMuxLinks(3)	Number of streams multiplexor links	87
streamUse(4)	Current number of open streams	15
streamMaxs(5)	Greatest number of open streams recorded	17
streamFails(6)	Number of stream allocation failures	0
queueUse(7)	Number of streams queues currently in use	54
queueMaxs(8)	Greatest number of open streams recorded	62
queueFails(9)	Number of streams queue allocation failures	0
mblockUse(10)	Number of streams message blocks in use	26
mblockMaxs(11)	Greatest number of message blocks in use at one time	187
mblockFails(12)	Number of streams message block allocation failures	0
dblockUse(13)	Number of streams data blocks currently in use	26
dblockMaxs(14)	Greatest number of data blocks in use at one time	187
dblockFails(15)	Number of streams data block allocation failures	0

User Information

Use the User table to retrieve information about the user accounts that have been created on the system. Each row in this table represents a user account and contains columns that represent the following:

- User's login name
- Password
- User ID
- Group ID
- User name
- User home directory
- Login shell

NOTE

Depending on your local security policies, you may want to disable support for this table. For more information about disabling this support, refer to “Configuring Support for User and Group Information” on page 133.

Figure 5 shows the user account information in a sample User table.

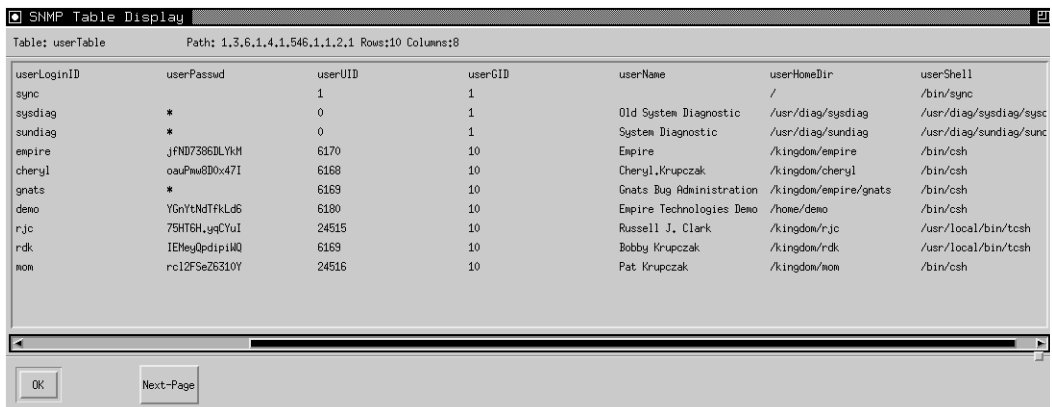


Table: userTable Path: 1.3.6.1.4.1.1546.1.1.2.1 Rows:10 Columns:8

userLoginID	userPasswd	userUID	userGID	userName	userHomeDir	userShell
sync		1	1		/	/bin/sync
sysdiag	*	0	1	Old System Diagnostic	/usr/diag/sysdiag	/usr/diag/sysdiag/sysc
sundiag	*	0	1	System Diagnostic	/usr/diag/sundiag	/usr/diag/sundiag/sunc
empire	jFN07386DLyKM	6170	10	Empire	/kingdom/empire	/bin/csh
cheryl	oauPmw6D0x47I	6168	10	Cheryl Krupczak	/kingdom/cheryl	/bin/csh
gnats	*	6169	10	Gnats Bug Administration	/kingdom/empire/gnats	/bin/csh
demo	YGnItNdTfKd6	6180	10	Empire Technologies Demo	/home/demo	/bin/csh
rjc	75HT6H.ygCYuI	24515	10	Russell J. Clark	/kingdom/rjc	/usr/local/bin/tcsh
rdk	IEfeyQdpipMQ	6169	10	Bobby Krupczak	/kingdom/rdk	/usr/local/bin/tcsh
nom	rc12Fse26310Y	24516	10	Pat Krupczak	/kingdom/nom	/bin/csh

OK Next-Page

Figure 5. Systems Management MIB: User Table

Group Information


Use the Group table to retrieve information about the user groups that have been created on the host system. Each row in the table represents a group defined in the `/etc/group` file, including the following information for each group:

- Group name
- Group password
- Group ID

NOTE

Depending on your local security policies, you may want to disable support for this table. For more information, refer to “Configuring Support for User and Group Information” on page 133.

Figure 6 shows a sample Group table.



The image shows a window titled "SNMP Table Display". Inside, it shows a table for "groupTable" with the path "1.3.6.1.4.1.546.1.1.3.1". The table has 10 rows and 4 columns. The columns are labeled "Instance", "groupIndex", "groupName", "groupPasswd", and "groupGID". The data rows are as follows:

Instance	groupIndex	groupName	groupPasswd	groupGID
1	1	wheel	*	0
2	2	nogroup	*	65534
3	3	daemon	*	1
4	4	kmem	*	2
5	5	bin	*	3
6	6	tty	*	4
7	7	operator	*	5
8	8	news	*	6
9	9	uucp	*	8
10	10	audit	*	9

At the bottom of the window, there are two buttons: "OK" and "Next-Page".

Figure 6. Systems Management MIB: Group Table

Process Information

Use the Process table to determine what processes are currently running on the system. Each row in the table represents a process and contains columns that represent the following:

- Process ID
- Name
- State
- Flags
- Owner user ID (UID)
- Owner group ID (GID)
- Scheduling priority
- Amount of memory and CPU time that the process is using

You can also control processes through the Process table. For example, you can increase or decrease a process's scheduling priority by setting the desired priority value in the process's `nice` column. You can also send a signal to a process by setting the `kill` column to the desired signal value. For example, from a remote network management station console, you can terminate an unauthorized process by setting the value of its **`processKill`** column to 9, the number that represents the UNIX SIGKILL signal, which can kill a process. Figure 7 shows a sample Process table.

SNMP Table Display

Table: processTable Path: 1.3.6.1.4.1.546.1.1.4.1 Rows:10 Columns:11

Instance	processID	processName	processState	processNice	processFlags	processUID	processGID	processKill	processSize	processRSS	processTime
128	128	lmgrd	1	20	537427968	0	0	0	72	0	0
129	129	screenblank	1	20	524288	0	0	0	16	0	1
139	139	islandLIC	1	20	536903680	0	0	0	132	0	1
142	142	rpc.yppasswdd	1	20	557056	0	0	0	44	0	0
147	147	update	1	20	524801	0	0	0	12	8	244
150	150	cron	1	20	4751360	0	0	0	56	0	0
156	156	inetd	1	20	557056	0	0	0	52	0	2
160	160	lpd	1	20	557056	0	1	0	52	0	0
331	331	csh	1	20	541622784	6168	10	0	68	0	0
398	398	snmpd	3	20	541622273	0	1	0	260	232	28

OK Next-Page

Figure 7. Systems Management MIB: Process Table

Changing the nice Value of a Process (UNIX only)

This section explains how to change the nice value for a specific process.

To change the nice value of a process:

1. Find the row that represents the selected process.
2. Set the **processNice** column in that row to the desired **nice** value (priority). For a list of values, refer to the UNIX `nice(1)` man page.

Sending a Signal to a Process

This section explains how to send a signal to a process.

When you are performing SNMP Set operations, use a community name that allows you read-write access.

To send a signal to a process:

1. Find the row that represents the selected process.
2. Set the **processKill** column in that row to the number corresponding to the desired signal. For more information, refer to the UNIX `signal(3V)` man page.

Who Table Information

Use the Who table to find out which users are currently logged on to the host system. Each row in this table represents a current user and contains columns that represents the following:

- User's name
- Login device
- Login PID
- Login time
- Location from where the user is logged in

The Who table can help you monitor who is using this system at a particular time.

NOTE

Depending on your local security policies, you may want to disable support for this table. For more information about disabling this support, refer to “Configuring Support for Who Table Information” on page 132.

Figure 8 shows a sample Who table, which lists the users who are currently logged on.

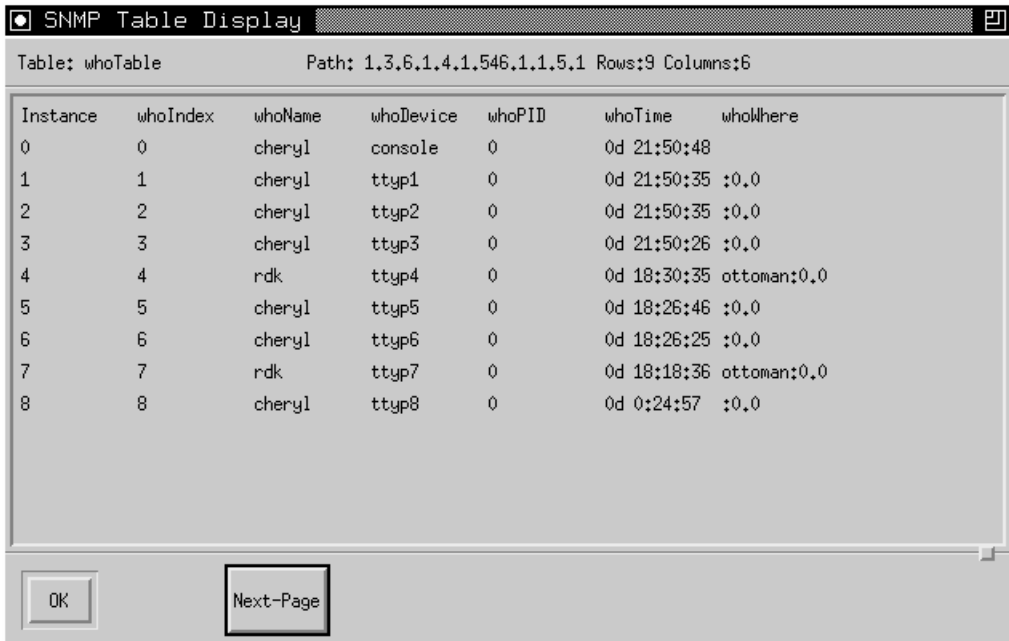


Table: whoTable Path: 1.3.6.1.4.1.546.1.1.5.1 Rows:9 Columns:6

Instance	whoIndex	whoName	whoDevice	whoPID	whoTime	whoWhere
0	0	cheryl	console	0	0d 21:50:48	
1	1	cheryl	ttyp1	0	0d 21:50:35	:0,0
2	2	cheryl	ttyp2	0	0d 21:50:35	:0,0
3	3	cheryl	ttyp3	0	0d 21:50:26	:0,0
4	4	rdk	ttyp4	0	0d 18:30:35	ottoman:0,0
5	5	cheryl	ttyp5	0	0d 18:26:46	:0,0
6	6	cheryl	ttyp6	0	0d 18:26:25	:0,0
7	7	rdk	ttyp7	0	0d 18:18:36	ottoman:0,0
8	8	cheryl	ttyp8	0	0d 0:24:57	:0,0

OK Next-Page

Figure 8. Systems Management MIB: Sample Who Table

Remote Command Execution

Use the Remote Shell Group to execute shell scripts and programs on the remote host system. MIB variables in the Remote Shell group let you specify the command, its arguments, and the name of a file where the output will be written. You can specify the command and arguments by setting the remoteShell MIB variable; specify the output file by setting shellOutput.

When you perform an SNMP Set on the shellCmd MIB variable, the agent *fork/execs* the specified command with standard output (stdout) and standard error (stderr) redirected to the file that is named by the shellOutput MIB variable. When the command finishes executing, the agent puts the command's exit status in shellExitStat.

Executing a Remote Command

When you are performing SNMP Set operations, use a community name that allows you read-write access.

Enter the following at a command prompt to execute a remote command:

```
SET shellOutput.0 = output filename
```

```
SET shellCmd.0 = command
```

NOTE

Depending on your local security policies, you may want to disable support for this table. For more information about disabling this support, refer to “Configuring Support for Remote Shell Capability” on page 133.

Kernel Performance Statistics

Use the Kernel Performance group to track the health and performance of the host’s operating system. Statistics that you can monitor include the following:

- Number of jobs that are waiting on disk I/O and page I/O
- Number of jobs in the scheduler’s run queue
- Number of active jobs that are swapped out, and the number that are sleeping
- Number of current processes and open files
- Statistics on paging, context switching, interrupts, and page faults

This group also includes a running percentage over 1-, 5-, and 15-minute intervals of the time that the processor was *not* in an idle state. For example, you can detect that the host system is overloaded if the 15-minute average is continuously high, and you can detect peak periods of CPU utilization by monitoring the 1-minute and 5-minute averages.

The following are the Kernel Performance Statistics MIB objects:

- cpu1Min(1)
- cpu5Min(2)

- cpu15Min(3)
- runQLen(4)
- diskWaitNum(5)
- pageWaitNum(6)
- swapActive(7)
- sleepActive(8)
- memInUse(9)
- activeMem(10)
- numProcs(11)
- numOpenFiles(12)
- swapInUse(13)
- numSwitches(14)
- numTraps(15)
- numSyscalls(16)
- numInterrupts(17)
- numPageSwapIn(18)
- numPageSwapOut(19)
- numSwapIn(20)
- numSwapOut(21)
- numPageIn(22)
- numPageOut(23)
- numPageReclaims(24)
- numPageFaults(25)
- loadAverage1Min(26)
- loadAverage5Min(27)
- loadAverage15Min(28)
- totalSwapSpace(29)
- swapCapacity(30)
- memCapacity(31)
- memInUseCapacity(32)
- pageScans(33)

For descriptions of these MIB objects, refer to the `empire.asn1` file in the `/doc` subdirectory of the SystemEDGE agent distribution.

Interprocess Communication: Queues, Shared Memory, and Semaphores

A semaphore is a value in operating system or kernel storage that a process can check and change to coordinate activities in which multiple process compete for the same operating system resources.

You can track the IPC mechanisms that are in use on the system by retrieving the appropriate MIB table:

- Message Queue table
- Shared Memory table
- Semaphore table

A row in each of these tables represents one instance of IPC usage. The columns differ for each table. The columns of the Message Queue table provide the message queue ID, key, mode, owner, group, and size. The columns of the Shared Memory table provide the shared memory segment ID, key, mode, owner, group, and size. The columns of the Semaphore table provide the semaphore ID, key, mode, and owner. Figure 9 shows a sample Shared Memory table.

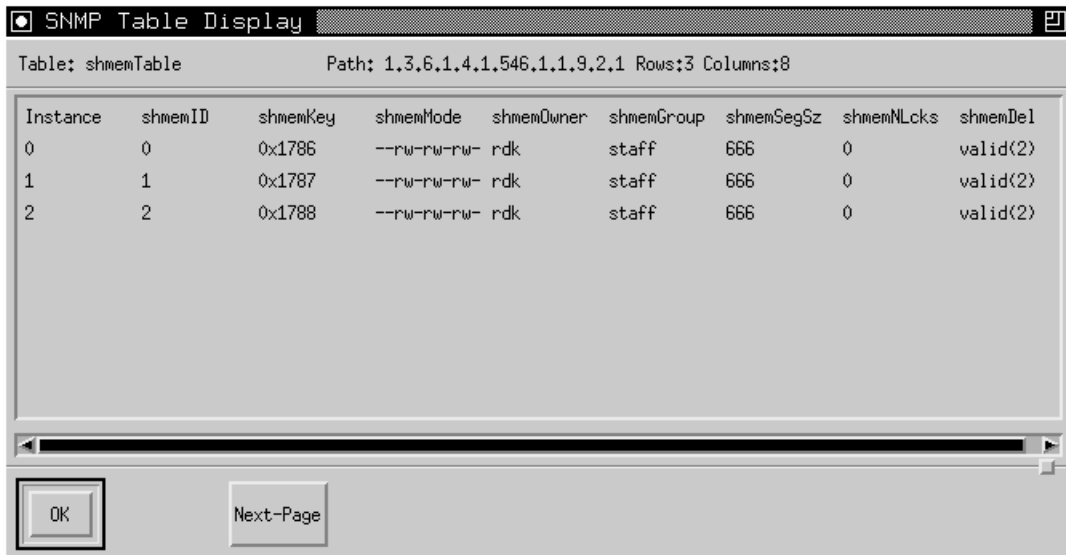


Table: shmemTable Path: 1.3.6.1.4.1.546.1.1.9.2.1 Rows:3 Columns:8

Instance	shmemID	shmemKey	shmemMode	shmemOwner	shmemGroup	shmemSegSz	shmemNLcks	shmemDel
0	0	0x1786	--rw-rw-rw-	rdk	staff	666	0	valid(2)
1	1	0x1787	--rw-rw-rw-	rdk	staff	666	0	valid(2)
2	2	0x1788	--rw-rw-rw-	rdk	staff	666	0	valid(2)

OK Next-Page

Figure 9. Systems Management MIB: Shared Memory Table

Deleting an Interprocess Communication

When you are performing SNMP Set operations, use a community name that allows you read-write access.

The IPC tables provide you with the power to delete message queues, shared memory segments, and semaphores using an SNMP Set operation. Each of these tables contains a column that acts as a Delete button; setting the column to the value of delete(1) causes the agent to destroy that instance of IPC usage.

The Delete button objects are named queDel (for the Message Queue table), shmemDel (for the Shared Memory table), and semDel (for the Semaphore table).

Message Buffer Allocation and Usage Statistics

Use the Message Buffer Allocation table to discover how your system is using message buffers (mbufs) and how many buffers have been allocated for each use. In addition, you can obtain statistics for the number of times mbuf requests were denied or delayed, which can help you track down message buffer shortages. Table 19 describes the MIB objects of the Message Buffer Group.

Table 19. Message Buffer Group MIB Objects

MIB Object	Description	Sample Value
numMbufs(1)	Total number of message buffers contained in the message buffer pool	608
numClusters(2)	Total number of logical pages or clusters obtained from the page pool	28
freeClusters(3)	Number of free clusters	28
numDrops(4)	Number of times requests for message buffers were denied	0
numWait(5)	Number of times requests for message buffers were delayed	0
numDrain(6)	Number of calls to protocol drain routines	0

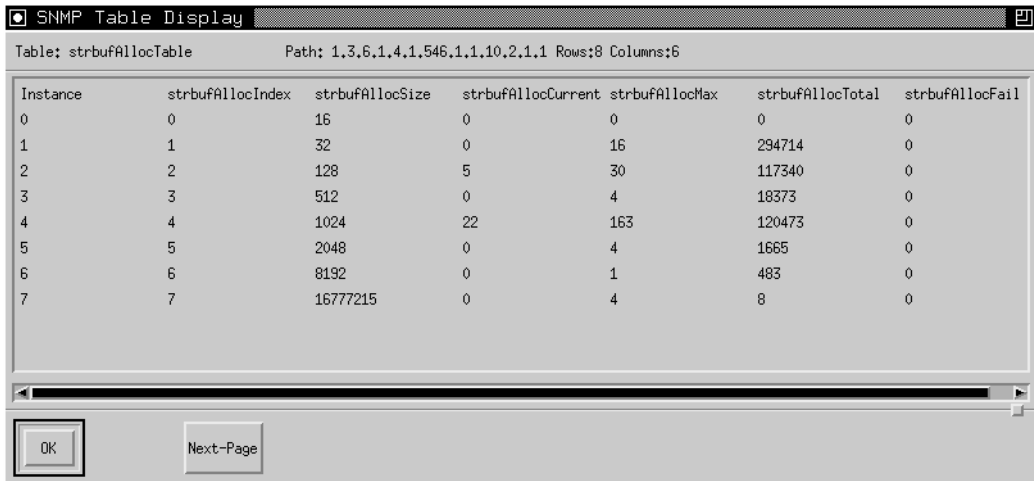
Figure 10 shows a sample Message Buffer Allocation table.

Instance	mbufType	mbufDesc	mbufAlloc
0	0	Free list	81
1	1	Dynamic (data) allocation	1
2	2	Packet header	6
3	3	Socket structure	165
4	4	Protocol control block	203
5	5	Routing tables	3
6	6	IMP host tables	0
7	7	Address resolution tables	0
8	8	Socket name	18
9	9	Zombie proc status	0

Figure 10. Systems Management MIB: Sample Message Buffer Allocation Table

Stream Buffers

Use the Streams Buffer Allocation table to monitor buffer allocation and usage statistics for buffers that are used by the Streams subsystem. Figure 11 shows a sample Streams Buffer Allocation table.



The image shows a window titled "SNMP Table Display". Inside, it displays the "strbufAllocTable" with the path "1.3.6.1.4.1.546.1.1.10.2.1.1". The table has 8 rows and 6 columns. The columns are labeled: Instance, strbufAllocIndex, strbufAllocSize, strbufAllocCurrent, strbufAllocMax, strbufAllocTotal, and strbufAllocFail. The data is as follows:

Instance	strbufAllocIndex	strbufAllocSize	strbufAllocCurrent	strbufAllocMax	strbufAllocTotal	strbufAllocFail
0	0	16	0	0	0	0
1	1	32	0	16	294714	0
2	2	128	5	30	117340	0
3	3	512	0	4	18373	0
4	4	1024	22	163	120473	0
5	5	2048	0	4	1665	0
6	6	8192	0	1	483	0
7	7	16777215	0	4	8	0

At the bottom of the window, there are two buttons: "OK" and "Next-Page".

Figure 11. Systems Management MIB: Sample Streams Buffer Allocation Table

I/O Buffer Cache

Use the I/O Buffer Cache group to track I/O buffer allocation and usage for basic disk I/O. You can also graph the values of these counters to detect peak periods of I/O buffer activity. Table 20 describes the I/O Buffer Cache group MIB objects.

Table 20. I/O Buffer Cache Group MIB Objects (Page 1 of 2)

MIB Object	Description	Sample Value
numBreadRequests(1)	Total number of buffer read calls that were made	1121570
numBreadHits(2)	Number of kernel buffer cache hits	1047985
numBufSleeps(3)	Total number of times kernel had to sleep for a buffer	0
numAgeAlloc(4)	Total number of times an aged buffer was allocated	15423

Table 20. I/O Buffer Cache Group MIB Objects (Page 2 of 2)

MIB Object	Description	Sample Value
numLRUAlloc(5)	Total number of times an LRU buffer was allocated	88217
minNumBufHdrs(6)	Minimum number of buffer headers allocated	30
numAllocBuf(7)	Current number of allocated buffers	30
ioBufferHitRate(8)	Percentage of read requests that are found in the buffer cache	90

RPC Group

Use the Remote Procedure Call (RPC) Group to track statistics and counters that relate to the use of the kernel's RPC facilities. You can graph the values of these counters to detect peak periods of RPC activity. Statistics and counters are divided according to their applicability towards client and server side protocol operations. For more information on RPC, refer to RFC 1057. The following are the RPC group MIB objects:

- clientRPCCalls(1)
- clientRPCBadcalls(2)
- clientRPCRetrans(3)
- clientRPCBadxid(4)
- clientRPCTimeout(5)
- clientRPCWait(6)
- clientRPCNewcred(7)
- clientRPCTimers(8)
- serverRPCCalls(9)
- serverRPCBadcalls(10)
- serverRPCNullrecv(11)
- serverRPCBadlen(12)
- serverRPCXdrcall(13)

For descriptions of these MIB objects, refer to the `empire.asn1` file in the `/doc` subdirectory of the SystemEDGE agent distribution.

NFS Group

Use the NFS group to track statistics and counters that are related to the use of the kernel's NFS facilities. You can graph the values of these counters to detect peak periods of NFS activity. Statistics and counters are divided according to their applicability towards client and server side protocol operations. For more information on NFS, refer to RFC 1094. The following are the NFS group MIB objects:

- `clientNFSCalls(1)`
- `clientNFSBadcalls(2)`
- `clientNFSNclgets(3)`
- `clientNFSNclsleeps(4)`
- `clientNFSNulls(5)`
- `clientNFSGetattrs(6)`
- `clientNFSSetattrs(7)`
- `clientNFSRoots(8)`
- `clientNFSLookups(9)`
- `clientNFSReadlinks(10)`
- `clientNFSReads(11)`
- `clientNFSWrcaches(12)`
- `clientNFSWrites(13)`
- `clientNFSCreates(14)`
- `clientNFSRemoves(15)`
- `clientNFSRenames(16)`
- `clientNFSLinks(17)`
- `clientNFSSymlinks(18)`
- `clientNFSMkdirs(19)`

- clientNFSRmdirs(20)
- clientNFSReaddir(21)
- clientNFSFsstats(22)
- serverNFSCalls(23)
- serverNFSBadcalls(24)
- serverNFSNulls(25)
- serverNFSGetattr(26)
- serverNFSSetattr(27)
- serverNFSRoots(28)
- serverNFSLookups(29)
- serverNFSReadlinks(30)
- serverNFSReads(31)
- serverNFSWrcaches(32)
- serverNFSWrites(33)
- serverNFSCreates(34)
- serverNFSRemoves(35)
- serverNFSRenames(36)
- serverNFSLinks(37)
- serverNFSSymlinks(38)
- serverNFSMkdirs(39)
- serverNFSRmdirs(40)
- serverNFSReaddir(41)
- serverNFSFsstats(42)

For descriptions of these MIB objects, refer to the `empire.asn1` file in the `/doc` subdirectory of the SystemEDGE agent distribution.

Windows-Specific Groups

In most cases, the SystemEDGE agent supports the same MIB objects for Windows and UNIX. However, the underlying Windows operating system does not support some of the MIB objects, and therefore cannot be implemented by the SystemEDGE agent for Windows. For a list of the MIB objects that are not supported by these Windows operating systems, refer to “Unsupported MIB Objects on Windows” on page 199. The following sections define groups that have been specifically designed for Windows systems (including Windows NT 4.0, Windows 2000, and Windows XP).

NT System Group

Use the NT System group to determine the following:

- Operating system version, build, and service-pack numbers
- Kernel configuration parameters
- Cluster information
- Other pertinent system information about the Windows system

Following are the NT System group MIB objects:

- ntSystemVersion(1)
- ntBuildNumber(2)
- ntServicePackNumber(3)
- ntWorkstationOrServer(4)
- ntfsDisable8dot3NameCreation(5)
- ntWin31FileSystem(6)
- ntCriticalSectTimeout(7)
- ntGlobalFlag(8)
- ntIoPageLockLimit(9)
- ntLargeSystemCache(10)
- ntPagedPoolSize(11)
- ntNonPagedPoolSize(12)

- ntPagingFiles(13)
- ntSystemPages(14)
- ntOptionalSubsystem(15)
- ntCmdlineOptions(16)
- ntLPTTimeout(17)
- ntDosMemSize(18)
- ntWowCmdline(19)
- ntWowSize(20)
- ntUserFullScreen(21)
- ntHistoryBufferSize(22)
- ntNumberHistoryBuffers(23)
- ntQuickEdit(24)
- ntScreenBufferSize(25)
- ntWindowSize(26)
- ntWindowsAppInitDLLs(27)
- ntWindowsDeviceNotSelectedTimeout(28)
- ntWindowsSpooler(29)
- ntWindowsSwapDisk(30)
- ntWindowsXmitRetryTimeout(31)
- ntSystemRoot(32)
- ntBuildType(33)
- ntSysStartOptions(34)
- ntSysBiosDate(35)
- ntSysBiosVersion(36)
- ntVideoResolution(37)
- ntCrashDumpEnabled(38)
- ntLogEvent(39)
- ntOverwrite(40)
- ntSendAlert(41)
- ntIsClustered(42)

- ntClusterName(43)
- ntClusterMembers(44)
- ntClusterIsActive(45)
- ntClusterActiveNode(46)

For descriptions of these MIB objects, refer to the `empire.asn1` file in the `/doc` subdirectory of the SystemEDGE agent distribution.

NT Thread Group

In the Windows operating system, the kernel schedules threads to run on the processor(s). A **thread** is a part of a process that executes program code. A process may contain one or many threads.

Use the NT Thread table to obtain detailed status information for each thread that is executing on the system including the following:

- Process to which the thread belongs
- Number of seconds the thread has been running
- Percentage of time that the thread has run in user and privileged modes
- Current state of the thread (ready, running, wait, terminated, and so on)
- Reason a thread is waiting if it is in a wait state

Table 21 describes the NT Thread group MIB objects.

Table 21. NT Thread Group MIB Objects (Page 1 of 2)

MIB Object	Description
ntThreadPID	Process to which the thread belongs
ntThreadNumber ID	Number for the thread within the process
ntThreadPrivTime	Percent of time that the thread was executing in privileged mode

Table 21. NT Thread Group MIB Objects (Page 2 of 2)

MIB Object	Description
ntThreadProcTime	Percent of time that the thread was using processor to execute commands
ntThreadUserTime	Percent of time that the thread was executing in user mode
ntThreadContextSwitches	Number of context switches
ntThreadElapsedTime	Number of seconds the thread has been running
ntThreadPriorityBase	Base priority level
ntThreadPriority	Current priority level
ntThreadWaitReason	Reason that the thread is waiting: executive, free page, page in, user request, virtual memory, and so on
ntThreadStartAddr	Starting virtual address
ntThreadState	Current state: ready, running, wait, terminated, and so on
ntThreadID	System-wide unique ID number for the thread

NT Registry Group

The Windows registry is a database repository for information about the system's configuration for hardware, user accounts, and applications.

The NT Registry group enables you to query the Windows registry. You can determine if your registry is in danger of growing larger than the size limit set for it by querying the following two objects:

- ntRegistryCurrentSize, which provides the current size of the registry in MB
- ntRegistrySizeLimit, which provides the size limit (in MB) that is defined for the registry

NT Service Group

Windows provides several programs that run as Windows services. The NT Service group provides detailed status information for each configured service on the system including the following:

- Service name
- Path to service executable
- Start type
- Parameters
- State
- Object name

Table 22 describes the NT Service group MIB objects.

Table 22. NT Service Group MIB Objects

MIB Object	Description
ntServiceIndex	Index of this service entry
ntServiceName	Name of the NT service
ntServicePathName	Path name of the executables
ntServiceStartType	Service start type (for example, Automatic)
ntServiceParameters	Start parameters
ntServiceState	Current state
ntServiceObjectName	Current user name that the service is using

NT System Performance Group

The NT System Performance group provides statistics that enable you to track the health and performance of the system's Windows operating system. Specifically, performance counters in the System Performance group let you see the following:

- How much time the processor(s) spent in user and privileged modes
- Number of context switches that have taken place
- Number of system calls and interrupts
- Number of jobs in the scheduler's run queue
- Performance counters for file system operations, such as the number of read, write, and control operations, and the total number of KB for each

Figure 12 shows the organization of the NT System Performance group.

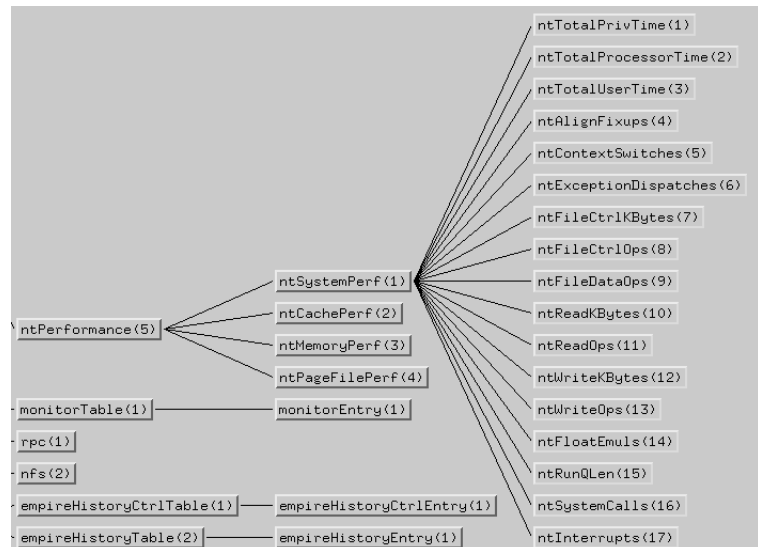


Figure 12. NT System Performance Group

Detecting a Heavily Loaded System

A heavily loaded system typically has a high level of system activity with many process threads competing for CPU time and jobs queueing up as they wait to run. You can track the level of system activity by monitoring the `ntContextSwitches` and `ntRunQLen` MIB objects.

`ntContextSwitches` counts the number of context switches that have occurred. A context switch occurs each time a thread gives up the CPU and another takes its place. A high rate of context switching indicates a high system load. `ntRunQLen` indicates whether there is a backup of threads waiting to run.

NT Cache Performance Group

The NT Cache Performance group contains system-wide buffer and filesystem cache statistics that let you determine the effectiveness of a system's caching mechanisms.

When an application requests data, the data is first mapped into the cache and then copied into memory from the cache. Later, data that is changed by the application is written from the cache back to disk by the Lazy Writer system thread or by a write-through call from the application.

Monitor the cache performance objects in this group to see if the cache is performing poorly (for example, if the system has a low cache hit ratio). If so, your system may need additional memory. Figure 13 shows the organization of the NT Cache Performance group.

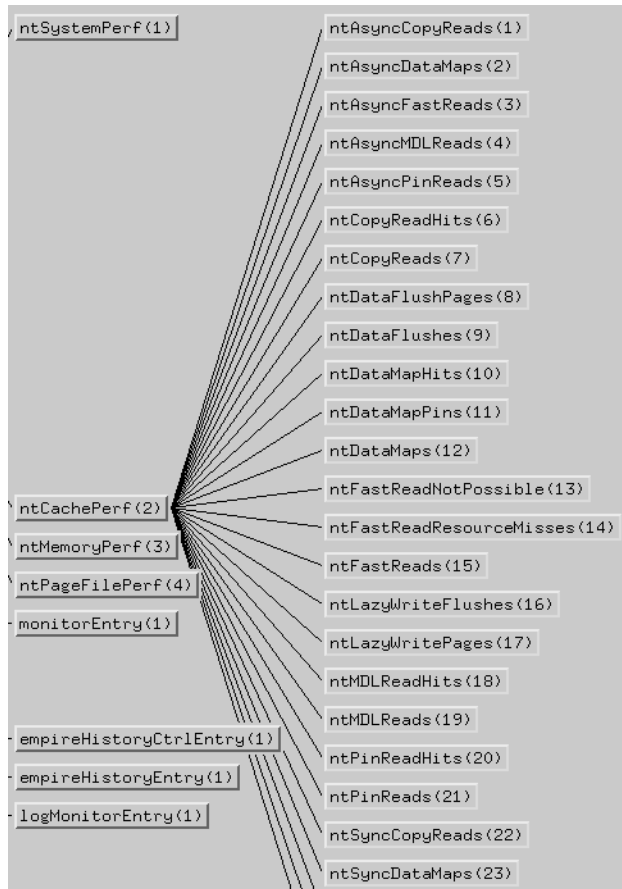


Figure 13. NT Cache Performance Group

NT Memory Performance Group

The NT Memory Performance group contains memory and paging performance counters and statistics that enable you to track the memory usage of your system.

You can determine, for example, the following:

- Amount of available virtual memory
- Number of KB being used by the system cache
- Number of page faults, page reads, and page writes
- Many other memory and paging-related statistics

For more information about the specific MIB objects, refer to the `empire.asn1` file in the `/doc` subdirectory of the SystemEDGE agent distribution. Figure 14 shows the organization of the NT Memory Performance Group.

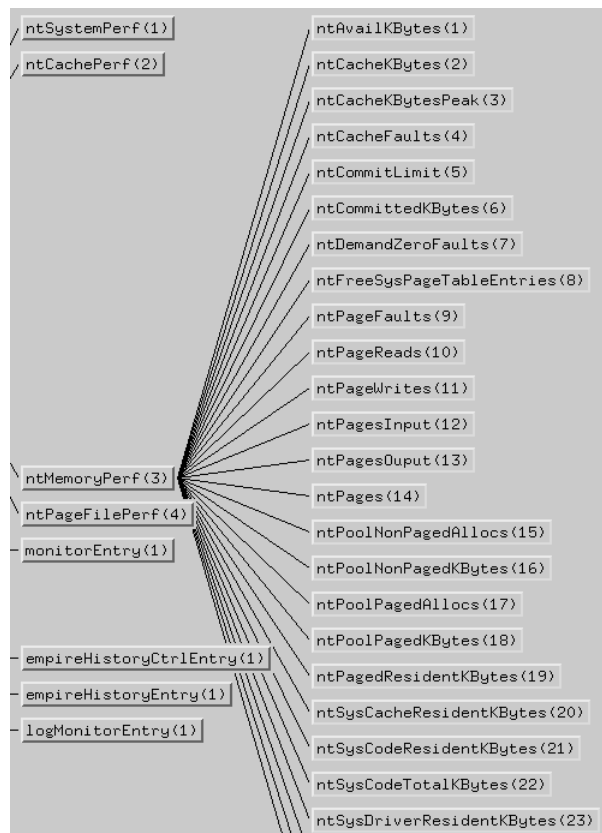


Figure 14. NT Memory Performance Group

NT Page File Performance Group

Windows uses the Pagefile.sys paging file to handle any extra demands for memory. The paging file is a block of disk space reserved by the operating system so that the memory manager can free space in memory when necessary. It does so by writing pages that are not often referenced to the paging file that is on disk. You can monitor the size of the paging file to determine whether you should add more memory to your system.

The NT Page File Performance group includes the following MIB objects:

- ntPageFileUsage, which provides the percentage of the page file space that is currently being used
- ntPageFilePeakUsage, which provides the maximum percentage of the page file(s) that were used when memory demand was at its peak

NT Event Monitor Group

Windows uses event logs to capture important system and application status messages. You can use the SystemEDGE agent Windows event monitoring capability to instruct the agent to continuously monitor Windows event logs for specific events that you specify. Whenever a matching event is generated on the system, the agent notifies the management systems with a trap message. The agent can also execute an action command to immediately handle the event.

The NT Event table provides detailed status information for each monitor entry. For more information about using Windows event monitoring, refer to Chapter 14, “Configuring Windows Event Monitoring.”

Table 23 describes the NT Event Monitor group MIB objects.

Table 23. NT Event Monitor Group MIB Objects

MIB Object	Description
ntEventMonIndex	Row Index
ntEventMonLog	Event Log to monitor
ntEventMonTime	Time of last matching event
ntEventMonMatches	Number of matches so far
ntEventMonTypeLastMatch	Event type of last match
ntEventMonTypeFilter	Event types to match
ntEventMonSrcLastMatch	Event source of last match
ntEventMonSrcFilter	Event source regular expression to match
ntEventLastMatch	Event description of the last match
ntEventMonDescFilter	Event description regular expression to match
ntEventMonStatus	Row status
ntEventMonDescr	Description
ntEventMonAction	Action to perform if a match is found
ntEventMonFlags	Flags that indicate additional behavior

NT Registry and Performance Extension Group

The SystemEDGE agent provides a powerful mechanism for extending the Systems Management MIB to include information from the Windows registry and performance counters. This includes both configuration data (normally viewed using regedit) and performance data (normally viewed using perfmon).

Using this feature, you can customize SystemEDGE to return additional configuration and performance information for systems and applications. For instance, many applications provide registry entries that specify the configuration of the application. The SystemEDGE agent can make these entries available through SNMP using the SystemEDGE agent. In addition, the SystemEDGE agent can also access statistics that many applications provide.

This support is provided in the ntRegPerfGroup of the Systems Management MIB. This group contains 128 unspecified scalar MIB variables that you can configure. In response to a SNMP Get request for one of these variables, the SystemEDGE agent reads the Windows Registry and returns the value. For more information about using and configuring MIB objects in the NT Registry Performance group, refer to Chapter 17, “Adding Windows Registry and Performance MIB Objects.”

Unsupported MIB Objects on Windows

The SystemEDGE agent for Windows supports the Systems Management MIB, but some of the MIB objects within this MIB module are not supported by the underlying Windows operating system. In addition, some of the UNIX-specific MIB objects are not applicable to Windows. Those objects, therefore, cannot be implemented by the Windows version of the agent.

The following Systems Management MIB objects are not supported by Windows:

- system.hostid
- devTable.devTfiles
- devTable.devFfiles
- devTable.devMaxNameLen
- devTable.devFstr
- kernelConfig.serialNumber
- kernelConfig.maxInode
- kernelConfig.maxFiles
- kernelConfig.maxClist
- kernelConfig.maxMemPerProc
- kernelConfig.openMaxPerProc
- kernelConfig.posixJobCtrl
- kernelConfig.posixVersion
- bootconf
- streams
- userTable.userUID
- userTable.userGID
- userTable.userShell
- processTable.processFlags
- processTable.processUID
- processTable.processGID
- processTable.processRSS
- processTable.processParentPID
- processTable.processInBlks
- processTable.processOutBlks
- processTable.processMsgsSent
- processTable.processMsgsRecv
- processTable.processSysCalls

- processTable.processMinorPgFaults
- processTable.processNumSwaps
- processTable.processVolCtx
- processTable.process.InvolCtx
- performance (except kernelperf)
- kernelperf.diskWaitNum
- kernelperf.pageWaitNum
- kernelperf.swapActive
- kernelperf.sleepActive
- kernelperf.numTraps
- kernelperf.numPageReclaims
- kernelperf.pageScans
- errorTable
- ipc
- buffers.mbuf
- buffers.strbuf
- ioBufferCache.numBufSleeps
- ioBufferCache.numAgeAllocs
- ioBufferCache.numLRUAllocs
- ioBufferCache.numBufHdrs
- ioBufferCache.numAllocBuff
- dnlc
- ntRegistry.ntRegistryCurrentSize
- rpc
- nfs

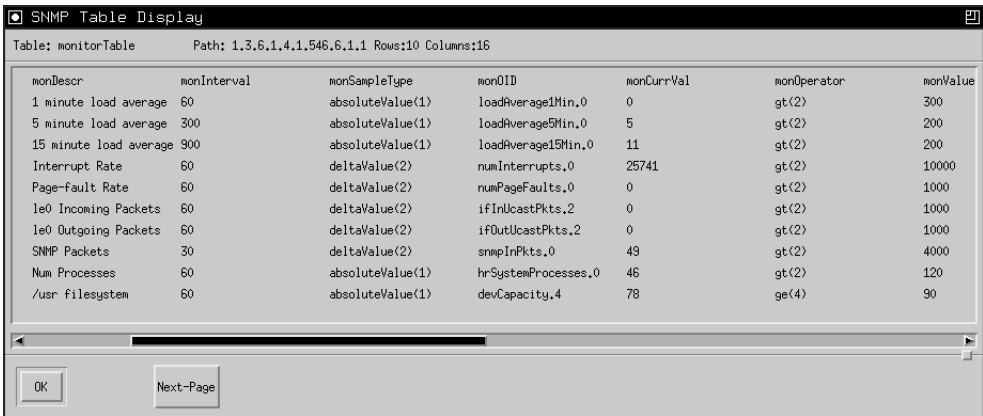
Monitor Table

The Monitor table enables you to dynamically configure the agent’s self-monitoring capability to monitor any integer-based MIB variable under its control; you select the polling interval, comparison operator (greater than, less than, equal to, and so on), and threshold value. The agent automatically monitors the MIB variable for you and notifies the management system with a trap message if an exception occurs.

The Monitor table allows you to specify an entry for any MIB variable within MIB-II, the Host Resources MIB, and the Systems Management MIB. As the agent is running, the entries in the Monitor table also show information such as the following:

- Time at which the variable specified by the entry was last sampled
- Value of the variable
- Lowest and highest values observed
- Number of times that a trap has been sent for the entry

Figure 15 shows examples of some conditions that you can configure the agent to monitor.



The image shows a window titled "SNMP Table Display" with a table titled "Table: monitorTable" and path "1.3.6.1.4.1.546.6.1.1". The table has 10 rows and 7 columns. The columns are: monDescr, monInterval, monSampleType, monOID, monCurrVal, monOperator, and monValue. The rows contain various system metrics and their current values and operators.

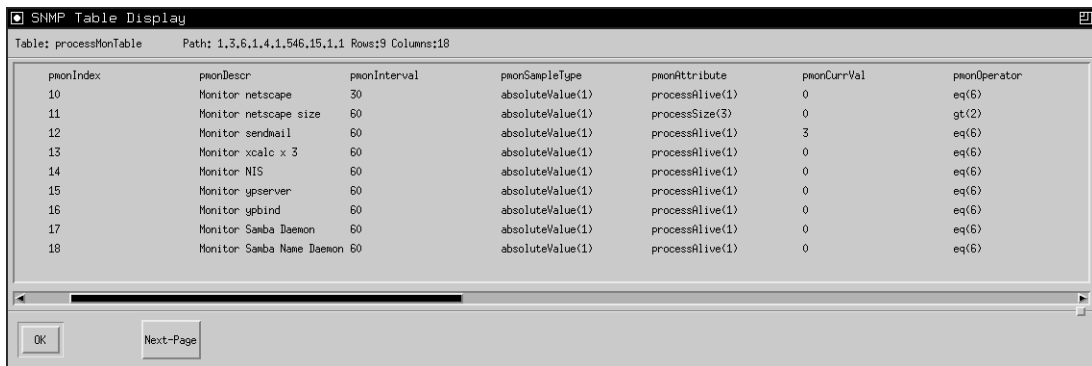
monDescr	monInterval	monSampleType	monOID	monCurrVal	monOperator	monValue
1 minute load average	60	absoluteValue(1)	loadAverage1Min,0	0	gt(2)	300
5 minute load average	300	absoluteValue(1)	loadAverage5Min,0	5	gt(2)	200
15 minute load average	900	absoluteValue(1)	loadAverage15Min,0	11	gt(2)	200
Interrupt Rate	60	deltaValue(2)	numInterrupts,0	25741	gt(2)	10000
Page-fault Rate	60	deltaValue(2)	numPageFaults,0	0	gt(2)	1000
le0 Incoming Packets	60	deltaValue(2)	ifInUcastPkts,2	0	gt(2)	1000
le0 Outgoing Packets	60	deltaValue(2)	ifOutUcastPkts,2	0	gt(2)	1000
SNMP Packets	30	deltaValue(2)	snmpInPkts,0	49	gt(2)	4000
Num Processes	60	absoluteValue(1)	hrSystemProcesses,0	46	gt(2)	120
/usr filesystem	60	absoluteValue(1)	devCapacity,4	78	ge(4)	90

Figure 15. Systems Management MIB: Sample Monitor Table

For more information about using the Monitor table and SystemEDGE threshold monitoring, refer to Chapter 10, “Configuring Threshold Monitoring.”

Process Monitor Table

The Process Monitor table provides a powerful and flexible mechanism for monitoring applications and processes. This capability allows you to monitor various attributes of key processes and applications and to send SNMP traps when certain thresholds have been exceeded. Figure 16 shows a sample Process Monitor table.



The image shows a window titled "SNMP Table Display" with a table titled "processMonTable". The table has 9 rows and 18 columns. The columns are: pmonIndex, pmonDescr, pmonInterval, pmonSampleType, pmonAttribute, pmonCurrVal, and pmonOperator. The table contains 9 rows of data, each representing a different process being monitored.

pmonIndex	pmonDescr	pmonInterval	pmonSampleType	pmonAttribute	pmonCurrVal	pmonOperator
10	Monitor netscape	30	absoluteValue(1)	processAlive(1)	0	eq(6)
11	Monitor netscape size	60	absoluteValue(1)	processSize(3)	0	gt(2)
12	Monitor sendmail	60	absoluteValue(1)	processAlive(1)	3	eq(6)
13	Monitor xcalc x 3	60	absoluteValue(1)	processAlive(1)	0	eq(6)
14	Monitor NIS	60	absoluteValue(1)	processAlive(1)	0	eq(6)
15	Monitor ypserver	60	absoluteValue(1)	processAlive(1)	0	eq(6)
16	Monitor ypbind	60	absoluteValue(1)	processAlive(1)	0	eq(6)
17	Monitor Samba Daemon	60	absoluteValue(1)	processAlive(1)	0	eq(6)
18	Monitor Samba Name Daemon	60	absoluteValue(1)	processAlive(1)	0	eq(6)

Figure 16. Systems Management MIB: Sample Process Monitor Table

For more information about using the Process Monitor table, refer to Chapter 11, “Configuring Process and Service Monitoring.”

Process Group Monitor Table

The Process Group Monitor table provides a powerful and flexible mechanism for monitoring groups of processes. Figure 17 shows a sample Process Group Monitor table.

Table: processGroupMonTable Path: 1.3.6.1.4.1.1.546.15.10.1 Rows:3 Columns:27

Instance	pgmonIndex	pgmonDescr	pgmonInterval	pgmonProcRegExpr	pgmonFlags	pgmonNumProcs
1	1	Watch NFS Server	60	nfsd	0	8
2	2	Watch Web Server	60	httpd	0	11
3	3	Watch SSH Server	60	sshd	0	2

OK Next-Page

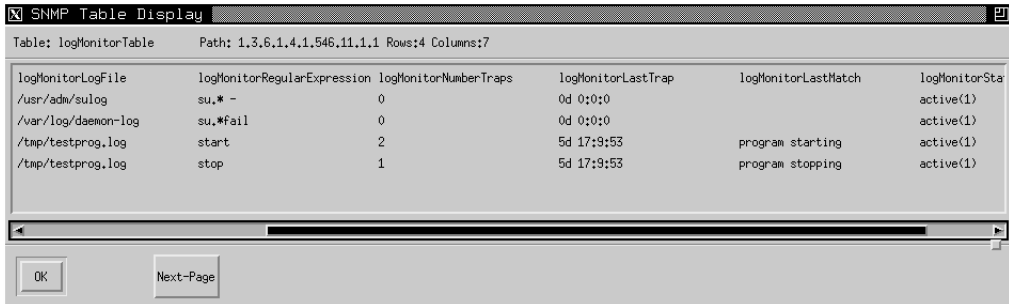
Figure 17. Systems Management MIB: Sample Process Group Monitor Table

For more information about using the Process Group Monitor table, refer to Chapter 12, “Configuring Process Group Monitoring.”

Log Monitor Table

The Log Monitor table enables you to dynamically configure the agent to monitor system log files for regular expressions. For example, you can configure the agent to monitor the `/var/adm/sulog` log file for the regular expression `su.*fail`. Whenever the agent finds a match, it sends an SNMP trap to the configured management systems. For more information about formatting the `logMonMatch` Trap PDU, refer to “`logMonMatch` Trap” on page 222.

Each row of the Log Monitor table represents the monitoring of a log file for a particular regular expression. Figure 18 shows example entries for the Log Monitoring table.



SNMP Table Display

Table: logMonitorTable Path: 1.3.6.1.4.1.546.11.1.1 Rows:4 Columns:7

logMonitorLogFile	logMonitorRegularExpression	logMonitorNumberTraps	logMonitorLastTrap	logMonitorLastMatch	logMonitorSta
/usr/adm/sulog	su.* -	0	0d 0:0:0		active(1)
/var/log/demon-log	su.*fail	0	0d 0:0:0		active(1)
/tmp/testprog.log	start	2	5d 17:9:53	program starting	active(1)
/tmp/testprog.log	stop	1	5d 17:9:53	program stopping	active(1)

OK Next-Page

Figure 18. Systems Management MIB: Sample Log Monitor Table

For more information about configuring SystemEDGE log file monitoring, refer to Chapter 13, “Configuring Log File Monitoring.”

History Table

The SystemEDGE agent can track the values of various integer-based MIB objects (counters, gauges, and so on) over time and store them for later retrieval. This functionality, commonly referred to as history sampling, can greatly reduce the amount of management station polling across the network. The agent provides this functionality through two SNMP MIB tables:

- History Control table
- History table

The History Control table contains parameters that describe the data that will be sampled and stored in the History table. Each row in the control table defines a specific data-collection function by assigning values to the parameters (column objects) of the table. One or more rows (stored samples) in the History table are associated with that single control row.

Each control table row is assigned a unique index value (empireHistoryCtrlIndex). A row defines the data-collection function by specifying the object-instance to be sampled, how often to sample (in multiples of 30 seconds), and the number of samples to keep (buckets). Associated with each data-collection function (row of the control table) is a *set of rows* of the History table. Each row of the History table, also called a bucket, holds the value of the specified MIB object gathered during one sampling interval.

As each sampling interval occurs, the agent adds a new row to the History table with the same empireHistoryIndex value as other rows for this data-collection function. This new row corresponds to the single row in the History Control table and has an empireHistorySampleIndex value that is one greater than the SampleIndex of the previous sample.

History Sampling Examples

Table 24 shows sample entries in the History Control table.

Table 24. Sample History Control Table Entries

Index	Desc	Int	ObjID	Type	Bckts-Req	Bckts-Grant	Last-Call	Create-Time	Status
1	1 Minute CPU Load Avg.	60	loadAverage1Min.0	2	5	5	0d 6:30:00	0d 0:0:0	active(1)
2	Memory In Use	30	memInUse.0	2	10	10	0d 6:30:30	0d 0:0:0	active(1)
3	Swap Capacity	900	swapCapacity.0	2	48	48	0d 6:30:00	0d 2:00:00	active(1)

The History Control table uses the SNMPv2 Structure of Management Information (SMI) Row Status Textual Convention for adding and deleting rows. For more information about the conventions for row status, refer to Appendix E.

In this example (Table 24), the rows are defined as follows:

- Control row 1 causes the agent to poll the 1-minute load average every minute and to store the most recent 5 samples (5 minutes).
- Control row 2 causes the agent to poll the memoryInUse MIB variable every 30 seconds and to store the most recent 10 samples (5 minutes).
- Control row 3 causes the agent to poll the swapCapacity variable (percentage of swap that is in use) every 15 minutes and to store the most recent 48 samples. By storing 48 samples, a management system needs to upload these samples only once every 12 hours (that is, 48 samples at 15-minute intervals provides 12 hours of coverage).

Table 25 shows the history samples that were stored in the History table as a result of the sampling configuration defined in the sample History Control (Table 24) on page 206.

Table 25. History Table Stored Samples (Page 1 of 2)

Index	SampleIndex	StartTime	SampleTime	Value
1	387	0d 0:0:0	0d 6:26:00	2
1	388	0d 0:0:0	0d 6:27:00	0
1	389	0d 0:0:0	0d 6:28:00	1
1	390	0d 0:0:0	0d 6:29:00	2
1	391	0d 0:0:0	0d 6:30:00	2
2	772	0d 0:0:0	0d 6:26:00	32204
2	773	0d 0:0:0	0d 6:26:30	32213
:	:	:	:	:
:	:	:	:	:
2	781	0d 0:0:0	0d 6:30:30	32224
3	1	0d 2:0:0	0d 2:00:00	70

Table 25. History Table Stored Samples (Page 2 of 2)

Index	SampleIndex	StartTime	SampleTime	Value
3	2	0d 2:0:0	0d 2:15:00	64
3	3	0d 2:0:0	0d 2:30:00	66
:	:	:	:	:
:	:	:	:	:
3	19	0d 2:0:0	0d 6:30:00	80

For more information about using the History Control table, refer to Chapter 15, “Configuring History Collection.”

Disk Statistics Group

The Disk Statistics group provides disk I/O statistics. Each table entry provides the latest disk statistics for one disk. The agent periodically (every 60 seconds) checks the status of the system data structures for each disk and records the values in the table. Table 26 describes the Disk Statistics group MIB objects.

Table 26. Disk Statistics Group MIB Objects (Page 1 of 2)

MIB Object	Description
diskStatsIndex	Index in the Disk Statistics table
diskStatsQueueLength	Average number of operations waiting
diskStatsServiceTime	Average service time in milliseconds
diskStatsUtilization	Percentage utilization
diskStatsKBytesTransferred	Total KB transferred
diskStatsTransfers	Total number of transfers

Table 26. Disk Statistics Group MIB Objects (Page 2 of 2)

MIB Object	Description
diskStatsReads	Total number of read operations
diskStatsWrites	Total number of write operations
diskStatsHostmibDevTableIndex	Index of this disk in the Host Resources MIB Device table
diskStatsLastUpdate	Time of last stats update

When you are using the Disk Statistics table, refer to the **diskStatsLastUpdate** column for information about the time (in TimeTicks) that this row was last updated. This column lets you know whether the statistics have been updated since you last queried the table. The values for **diskStatsQueueLength**, **diskStatsServiceTime**, and **diskStatsUtilization** are all calculated over the interval between updates.

Enabling Collection of Disk-Performance Statistics

For Windows and AIX systems, you must manually configure the systems to collect disk-performance statistics. The SystemEDGE agent can monitor these statistics only if you instruct the operating system to track them.

NOTE

Enabling the collection of these statistics decreases the disk throughput.

Windows Systems

This section describes how to enable and disable the collection of disk statistics for Windows systems.

To enable collection of disk statistics for Windows systems:

1. Log in to the system you want to monitor as an administrator.
2. Enter the following at the command prompt:

```
diskperf -y
```

3. Restart the system.

To disable collection of disk statistics for Windows systems:

1. Log in to the system you want to monitor as an administrator.
2. Enter the following at the command prompt:

```
diskperf -n
```

3. Restart the system.

AIX Systems

This section describes how to enable and disable the collection of disk statistics for AIX systems.

To enable collection of disk statistics for AIX systems:

1. Log in to the system you want to monitor with root privileges.
2. Enter the following at the command prompt:

```
chdev -l sys0 -a iostat=true
```

To disable collection of disk statistics for AIX systems:

1. Log in to the system you want to monitor with root privileges.
2. Enter the following at the command prompt:

```
chdev -l sys0 -a iostat=false
```

CPU Statistics Group

The CPU Statistics group provides performance statistics for each CPU. Each table entry provides the latest statistics for one CPU. The agent periodically (every 60 seconds) checks the status of the system data structures for each CPU and records the values in the table. Table 27 describes the CPU Statistics MIB objects.

Table 27. CPU Statistics Group MIB Objects (Page 1 of 2)

MIB Object	Description
cpuStatsIndex	Index in this table
cpuStatsDescr	Textual description of CPU type
cpuStatsIdle	Total number of ticks spent in Idle mode
cpuStatsUser	Total number of ticks spent in User mode
cpuStatsSys	Total number of ticks spent in System mode
cpuStatsWait	Total number of ticks spent in Wait mode
cpuStatsLastUpdate	Time of last update
cpuStatsIdlePercent	Percentage of sample time spent in Idle mode

Table 27. CPU Statistics Group MIB Objects (Page 2 of 2)

MIB Object	Description
cpuStatsUserPercent	Percentage of sample time spent in User mode
cpuStatsSysPercent	Percentage of sample time spent in Sys mode
cpuStatsWaitPercent	Percentage of sample time spent in Wait mode

When you are using the CPU statistics table, refer to the **cpuStatsLastUpdate** column for information about the time (in TimeTicks) that this row was last updated. This column also lets you know whether the statistics have been updated since you last queried the table. The values for `cpuStatsIdlePercent`, `cpuStatsUserPercent`, `cpuStatsSysPercent`, and `cpuStatsWaitPercent` are all calculated over the interval between updates.

Extension Group

The Extension group provides a powerful mechanism for extending the Systems Management MIB to include a wide range of information on your systems and applications. This group contains 2^{32} unspecified scalar MIB variables that you can configure. In response to an SNMP Get request for one of these variables, the SystemEDGE agent invokes the command that you specify for the variable and returns the value that is returned from the command. Using an SNMP Set operation, you can also pass parameters to a command. Figure 19 shows the organization of a sample Extension group.

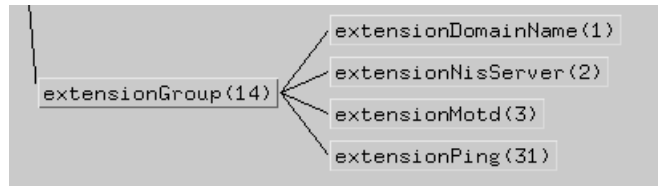


Figure 19. Systems Management MIB: Sample Extension Group Entries

For more information about using the Extension group variables, refer to Chapter 16, “Adding Custom MIB Objects.”

Private Enterprise Traps

In addition to supporting the standard MIB-II traps, the SystemEDGE agent supports a number of private-enterprise trap types that have been defined for use with the agent's self-monitoring capabilities. This chapter describes the format of the Trap PDUs that the SystemEDGE agent can send.

Format of Trap PDUs

The Systems Management MIB file, `empire.asn1`, defines enterprise-specific traps for use with the SystemEDGE agent. This section describes the PDU format, including the information that is contained in the variable-bindings fields, for each type of trap.

NOTE

Variable bindings are optional fields in a Trap PDU that associate a particular object instance with its current value.

The text in this chapter is taken from the Systems Management MIB definition, which exists in the `/doc` subdirectory of the SystemEDGE agent distribution.

All traps that are sent by the SystemEDGE agent contain the following enterprise system object identifier (`sysObjectID`) `empire(546).1.1`.

monitor Trap

The following code shows the format of a monitor trap, which the SystemEDGE agent sends to indicate that a monitor event has occurred. The agent sends this trap when the expression *monCurrVal monOperator monValue* evaluates to True.

```
monitorEvent OBJECT IDENTIFIER ::= { traps 1 }
```

```
monitorTrap TRAP-TYPE
```

```
ENTERPRISE empire
```

```
VARIABLES { monDescr, monOID, monCurrVal, monValue,  
monRowStatus, monOperator, monIndex, monFlags }
```

```
DESCRIPTION
```

```
"A Monitor event has occurred. Recall a  
monitor event occurs when a Monitor Table row  
expression evaluates to true. The expression  
is: 'monCurrVal monOperator monValue'."
```

```
::= 1
```


monitorEntryNotReady Trap

The following code shows the format of a monitorEntryNotReady trap, which the SystemEDGE agent sends to indicate that the monRowStatus field of a Monitor table entry is set to notReady(3).

```
monitorEntryNotReadyEvent OBJECT IDENTIFIER ::=
{ traps 3 }
```

```
monitorEntryNotReadyTrap TRAP-TYPE
```

```
ENTERPRISE empire
```

```
VARIABLES { monDescr, monOID, monCurrVal, monValue,
monRowStatus, monOperator, monIndex, monFlags }
```

DESCRIPTION

"This trap is sent when a Monitor Table entry's monRowStatus is set to 'notReady(3)'. One reason this may occur is that the object-instance identifier being monitored by its corresponding Monitor Table entry is no longer in existence. For example, when a process is being monitored (via the Empire processTable), and that process should exit, its corresponding entry in the Empire processTable would no longer exist. Consequently, the object-instance identifier (for that process) being monitored would no longer exist and the agent would send a monitorEntryNotReady trap to all properly configured managers."

```
::= 3
```

logMonMatch Trap

The following code shows the format of a logMonMatch trap, which the SystemEDGE agent sends to indicate that the log-file-monitoring subsystem has detected a match in a log file that the agent is currently monitoring.

```
logMonMatchEvent OBJECT IDENTIFIER ::= { traps 4 }

logMonMatchTrap TRAP-TYPE

ENTERPRISE empire

VARIABLES { logMonitorLogFile,
logMonitorRegularExpression,
logMonitorLastTrap, logMonitorLastMatch,
logMonitorDescr, logMonitorIndex, logMonitorFlags }

DESCRIPTION
"This trap is sent when the log monitoring
subsystem detects a match in a log file it is
currently monitoring. Periodically, the agent
stats each log file for changes; if any changes
have occurred, the agent scans only those changes
for a pattern match. Pattern matches result in
logMonMatch events. Changes to log files occur
when new entries are added by syslogd(1M) or other
logging daemons."
 ::= 4
```

logMonNotReady Trap

The following code shows the format of a logMonNotReady trap, which the SystemEDGE agent sends to indicate that the status of an entry in the Log Monitor table has changed to notReady.

```
logMonNotReadyEvent OBJECT IDENTIFIER ::= { traps 5 }
```

```
logMonNotReadyTrap TRAP-TYPE
```

```
ENTERPRISE empire
```

```
VARIABLES { logMonitorLogFile,
logMonitorRegularExpression,
logMonitorLastTrap, logMonitorLastMatch,
logMonitorDescr, logMonitorIndex, logMonitorFlags }
```

DESCRIPTION

"This trap is sent when the status of a log monitoring entry becomes 'notReady(3)'. An entry becomes 'notReady(3)' if an error occurs during log file scanning, if the regular expression is syntactically incorrect, or if the log file does not exist. An entry that is 'notReady(3)' will undergo no further evaluation until its status becomes 'active(1)'."

```
::= 5
```

ntEventMonMatch Trap

The following code shows the format of an ntEventMonMatch trap, which the SystemEDGE agent sends to indicate that the event-log-monitoring subsystem has detected a match in a Windows event log file that the agent is currently monitoring.

```
ntEventMonMatchEvent OBJECT IDENTIFIER ::=
{ traps 7 }

ntEventMonMatchTrap TRAP-TYPE

ENTERPRISE empire

VARIABLES { ntEventMonLog, ntEventMonTypeLastMatch,
ntEventMonTime, ntEventMonSrcLastMatch,
ntEventMonDescLastMatch, ntEventMonDescr,
ntEventMonIndex, ntEventMonFlags }

DESCRIPTION
"This trap is sent when the event log monitoring
subsystem detects a match in a log it is
currently monitoring. Periodically, the agent
checks each event log for changes; if any changes
have occurred, the agent scans only those changes
for a pattern match. Pattern matches result in
ntEventMonMatch events."

::= 7
```

ntEventMonNotReady Trap

The following code shows the format of an ntEventMonNotReady trap, which the SystemEDGE agent sends to indicate that the status of a Windows event log monitoring entry has become notReady(3).

```
ntEventMonNotReadyEvent OBJECT IDENTIFIER ::=
{ traps 8 }

ntEventMonNotReadyTrap TRAP-TYPE

ENTERPRISE empire

VARIABLES { ntEventMonLog, ntEventMonTypeFilter,
ntEventMonSrcFilter, ntEventMonDescFilter,
ntEventMonDescr, ntEventMonIndex, ntEventMonFlags }

DESCRIPTION
"This trap is sent when the status of an event log
monitoring entry becomes 'notReady(3)'. An entry
becomes 'notReady(3)' if an error occurs during log
file scanning, if the regular expression is
syntactically incorrect, or if the log file does
not exist. An entry that is 'notReady(3)' will
undergo no further evaluation until its status
becomes 'active(1)'."

::= 8
```

monitorClear Trap

The following code shows the format of a monitorClear trap, which the SystemEDGE agent sends to indicate that a condition that previously existed no longer exists. The SystemEDGE agent sends this trap when a Monitor table entry that has a set Clear Trap flag transitions from True to False.

```
monitorClearEvent OBJECT IDENTIFIER ::= { traps 9 }
```

```
monitorClearTrap TRAP-TYPE
```

```
ENTERPRISE sysmgmt
```

```
VARIABLES { monDescr, monOID, monCurrVal, monValue,  
monRowStatus, monOperator, monIndex, monFlags }
```

DESCRIPTION

"This trap is sent when a Monitor Table entry transitions from True to False and is using a clear trap flag. This trap indicates that the condition that previously had existed no longer does. This Trap provides management software the ability to determine that an alarm can be canceled or marked as corrected. This event only occurs when a monitor table entry evaluates to True and then evaluates to False. This Trap is sent each time the entry transitions from True to false."

```
::= 9
```

processStop Trap

The following code shows the format of a processStop trap, which the SystemEDGE agent sends to indicate that a process it was monitoring has either stopped running or is in a state where it cannot run.

```
processStopEvent OBJECT IDENTIFIER ::= { traps 10 }
```

```
processStopTrap TRAP-TYPE
```

```
ENTERPRISE sysmgmt
```

```
VARIABLES { pmonIndex, pmonDescr, pmonAttribute,  
pmonCurrVal, pmonOperator, pmonValue,  
pmonFlags, pmonRegExpr, pmonCurrentPID }
```

DESCRIPTION

"This Trap is sent when using a Process Monitor Table entry to monitor the state of a process. When the processing being monitored dies or transitions into a Zombie (or not runnable state), this Trap is sent. This Trap is sent if the value of pmonFlags does not preclude sending Traps."

```
::= 10
```

processStart Trap

The following code shows the format of a processStart trap, which the SystemEDGE agent sends to indicate that a process has restarted (and has been reacquired by the SystemEDGE agent).

```
processStartEvent OBJECT IDENTIFIER ::= { traps 11 }

processStartTrap TRAP-TYPE

ENTERPRISE sysmgmt
VARIABLES { pmonIndex, pmonDescr, pmonAttribute,
pmonCurrVal, pmonOperator, pmonValue,
pmonFlags, pmonRegExpr, pmonCurrentPID }

DESCRIPTION
"This Trap is sent when using a Process Monitor
Table entry to monitor the state of a process.
When a process is re-started, and subsequently
re-acquired by the SystemEDGE agent, this Trap is
sent. This Trap is sent if the value of pmonFlags
specifies that processStart Traps should be pay."

::= 11
```


processThreshold Trap

The following code shows the format of a processThreshold trap, which the SystemEDGE agent sends to indicate that an attribute of a process that it is monitoring has reached the specified threshold.

```
processThresholdEvent OBJECT IDENTIFIER ::=
{ traps 12 }

processThresholdTrap TRAP-TYPE

ENTERPRISE sysmgmt
VARIABLES { pmonIndex, pmonDescr, pmonAttribute,
pmonCurrVal, pmonOperator, pmonValue,
pmonFlags, pmonRegExpr, pmonCurrentPID }

DESCRIPTION
"This Trap is sent when using a Process Monitor
Table entry to monitor some attribute (e.g. memory
usage, process size) of a process for some
threshold. When a Process Monitor table
expression evaluates to True, this Trap is sent.
The expression is: 'pmonCurrVal pmonOperator
pmonValue'.
This Trap is sent if the value of pmonFlags does
not preclude the sending of Traps."

 ::= 12
```

processClear Trap

The following code shows the format of a processClear trap, which the SystemEDGE agent sends to indicate that a process attribute for which it previously sent a processThreshold event is no longer at the threshold level.

```
processClearEvent OBJECT IDENTIFIER ::= { traps 13 }
```

```
processClearTrap TRAP-TYPE
```

```
ENTERPRISE sysmgmt
```

```
VARIABLES { pmonIndex, pmonDescr, pmonAttribute,  
pmonCurrVal, pmonOperator, pmonValue,  
pmonFlags, pmonRegExpr, pmonCurrentPID }
```

DESCRIPTION

"This Trap is sent when using a Process Monitor Table entry to monitor some attribute (e.g. memory usage, process size) of a process for some threshold. When the threshold is crossed, a processThreshold Trap is sent. When the attribute threshold expression first transitions from True to False, this Trap is sent."

```
::= 13
```

license Trap

The following code shows the format of a license trap, which the SystemEDGE agent sends to indicate that it did not find a valid license for itself or one of the eHealth AIMs.

```
licenseEvent OBJECT IDENTIFIER ::= { traps 16 }
licenseTrap TRAP-TYPE
ENTERPRISE sysmgmt
VARIABLES { sysedgeLicenseString }
DESCRIPTION
"This Trap is sent when SystemEDGE or associated
modules failed to find a valid license. It can
be used in conjunction with auto-licensing or
remote-licensing starting with SystemEDGE 4.0.
This Trap contains a single MIB object
denoting which product or module failed to
find a valid license and a string containing
the license information for that product or module."
::= 16
```

addrChangeTrap

The following code shows the format of an addrChange trap, which the SystemEDGE agent sends to indicate that the underlying IP address has changed.

```
addrChangeEvent OBJECT IDENTIFIER ::= { traps 18 }
addrChangeTrap TRAP-TYPE
ENTERPRISE sysmgmt
VARIABLES { nodename sysedgeAddressList }
DESCRIPTION
"This Trap is sent when SystemEDGE detects
that its underlying IP address has changed perhaps
due to DHCP or other administrative means. It
includes up to the last 5 IP addresses that this
system was configured with. The addresses are
ordered with most recently used addresses occurring
first in the address list. This Trap may be used
on multi-homed systems."
::= 18
```

procGroupChangeTrap

The following code shows the format of a procGroupChange trap, which the SystemEDGE agent sends to indicate that the process group has changed.

```
procGroupChangeEvent OBJECT IDENTIFIER ::=
{ traps 19 }

procGroupChangeTrap TRAP-TYPE
ENTERPRISE sysmgmt
VARIABLES { pgmonIndex, pgmonDescr, pgmonFlags,
pgmonNumProcs, pgmonProcRegExpr, pgmonRowStatus,
pgmonPIDList, pgmonStatusList}
DESCRIPTION
"Membership in a process group has changed. (For
example, members have joined or left the group or
have changed.)"
::= 19
```

SNMP Trap Format

The SystemEDGE agent sends Trap PDUs to the SNMPv1 Trap port (UDP/162). Figure 20 shows the structure of an SNMPv1 Trap PDU.

SNMP Trap PDU

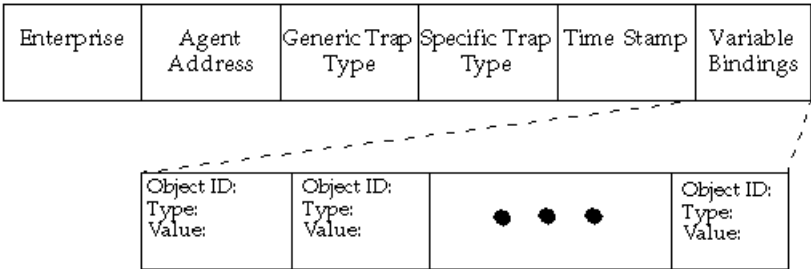


Figure 20. Structure of SNMP Trap PDU

Table 28 describes the components of the SNMP Trap PDU.

Table 28. Components of the SNMP Trap PDU (Page 1 of 2)

Component	Description
Enterprise	System Object ID of the sender: empire(546).1.1
Source address	IP address of sending host
Generic trap type	One of the following: <ul style="list-style-type: none">coldStart(0)warmStart(1)linkDown(2)linkUp(3)authenFailure(4)egpNeighborloss(5)enterpriseSpecific(6)

Table 28. Components of the SNMP Trap PDU (Page 2 of 2)

Component	Description
Specific trap type	Enterprise-specific trap type
Time stamp	Value of sysUptime when the trap was sent; always 0 for sendtrap
Variable bindings	Array of MIB variables and their values

Host Resources MIB

This chapter describes the management information that is available through the IETF Host Resources MIB (RFC 1514).

NOTE

The examples in this chapter do not describe the entire Host Resources MIB. For a description of the entire MIB, refer to the `hostmib.asn1` file, which is included in the SystemEDGE agent distribution.

The Host Resources MIB defines a set of objects that can manage host computers, which are independent of the operating system, network services, or any software application. The objects defined in the Host Resources MIB are common across many computer system architectures. Figure 21 shows the overall organization of the Host Resources MIB.

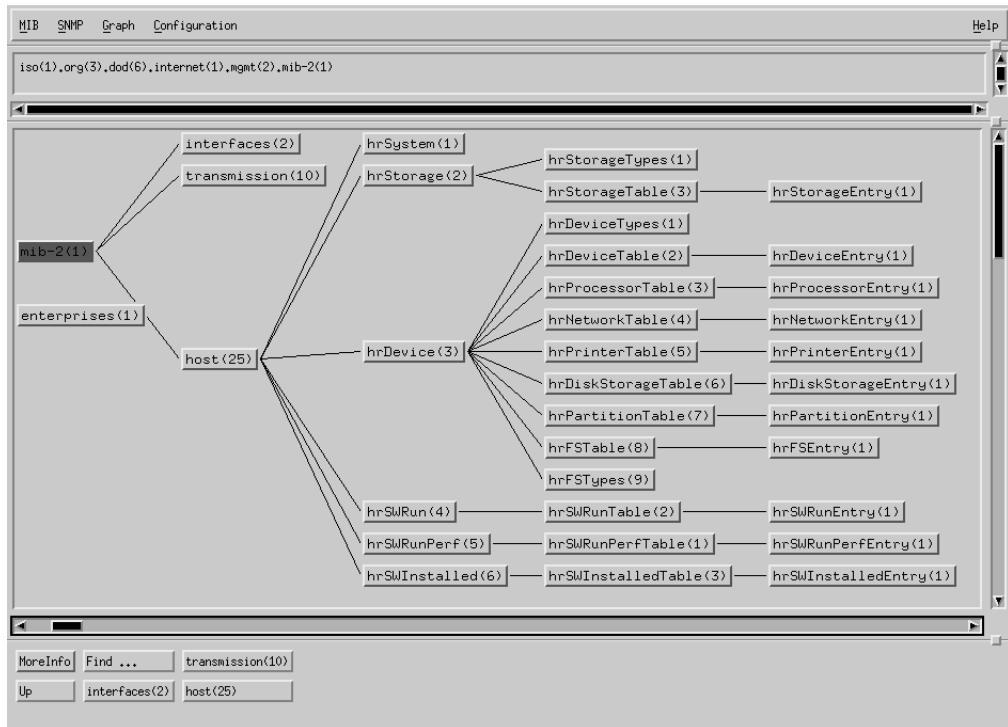


Figure 21. Host Resources MIB

Host Resources System Group

The Host Resources System (hrSystem) group provides you with information that pertains to the host system as a whole. Figure 22 shows the organization of the hrSystem group.

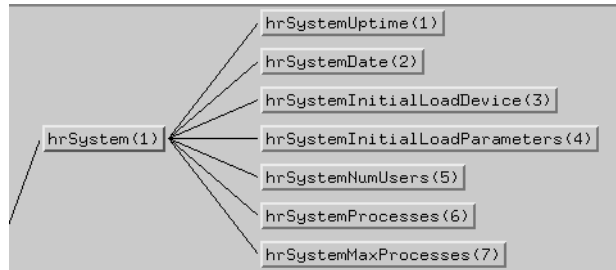


Figure 22. Host Resources System Group

Table 29 describes the information that is available through the Host Resources System group.

Table 29. MIB Objects: Host Resources System Group

MIB Object	Description
HrSystemUptime	Amount of time since the host was last initialized
HrSystemDate	Host's settings for the local date and time of day
HrSystemInitialLoadDevice	Device from which the host loads its initial operating system configuration
HrSystemInitialLoadParameters	Pathname and parameters applied when loading the initial operating system
HrSystemNumUsers	Number of user sessions for which the host is storing state information
HrSystemProcesses	Number of processes that are currently loaded or running on the system
HrSystemMaxProcesses	Maximum number of processes that can be run on this system

Host Resources Storage Group

The Host Resources Storage group (hrStorageTable) lists the logical areas of storage allocated on the host system. These storage areas include file systems and disk partitions that might be seen by an application, rather than physical storage such as tapes and floppy drives.

This table is intended to be a useful diagnostic for out-of-memory and out-of-buffers types of failures. In addition, it can be a useful performance-monitoring tool for tracking memory, disk, or buffer usage. Figure 23 shows the output of the Host Resources Storage group from a Sun SPARC IPX.

hrStorageType	hrStorageDescr	hrStorageAllocationUnits	hrStorageSize	hrStorageUsed	hrStorageAllocationFailures
hrStorageRam	Physical RAM	1	24801280	4567040	0
hrStorageVirtualMemory	Virtual Memory	1	103849984	0	0
hrStorageFixedDisk	Disk partition /dev/sd0a	1024	15487	11080	0
hrStorageFixedDisk	Disk partition /dev/sd0g	1024	160423	130426	0
hrStorageFixedDisk	Disk partition /dev/sd0h	1024	123911	29154	0

Figure 23. Sample Host Resources Storage Table

Host Resources Device Group

The Host Resources Device (hrDevice) group contains several tables that provide information about the devices that are contained by the host system. The main table in this group is the hrDeviceTable. It lists all of the devices that the host contains. The hrDevice group also includes a number of device-specific tables that provide more detailed information for particular device types.

For example, the group includes device-specific tables for the following:

- Processors
- Networks
- Printers
- Disk storage
- Partitions
- File systems

If the hrDeviceTable shows that the host contains a Printer device, for example, you can retrieve more detailed information about the printer from the hrPrinterTable. The following sections describe the hrDeviceTable and the device-specific tables.

Device Table

The hrDeviceTable lists the devices in the host system. For each device in the hrDeviceTable, the table lists the following:

- Device type
- Description of the device
- Status (for example, running or down)
- Number of errors detected for the device

Figure 24 shows an hrDeviceTable for a Sun SPARC IPX.

hrDeviceType	hrDeviceDescr	hrDeviceID	hrDeviceSt		
hrDeviceOther	SUNW,SPARCstation-LX, Sun Sparc Workstation	null	running(2)		
hrDeviceProcessor	TI,TMS390S10	null	running(2)		
hrDeviceOther	openprom, PROM monitor configuration interface	null	running(2)		
hrDeviceSerialPort	zs0, Zilog 8530 SCC Serial Communications Driver	null	running(2)		
hrDeviceSerialPort	zs1, Zilog 8530 SCC Serial Communications Driver	null	running(2)		
hrDeviceOther	sbus0, Main Memory and Bus I/O Space	null	running(2)		
hrDeviceParallelPort	bpp0, Bidirectional parallel port	null	running(2)		
hrDeviceNetwork	FORE,sba-2000, Fore ATM SBus Adapter	null	unknown(1)		
hrDeviceNetwork	SUNW,DBRIe0, Dual Basic Rate ISDN Interface and Speaker	null	unknown(1)		
hrDeviceVideo	ogsix0, Accelerated 8-bit Color Frame Buffer	null	running(2)		

Figure 24. Sample Host Resources Device Table

Inventory Tracking and Asset Management

The `hrDeviceTable` is especially useful for inventory tracking and asset management. Instead of manually opening and inspecting each workstation on your network to determine the number and types of ethernet cards, serial devices, audio devices, disk storage devices, and so on, you can use your central management system to retrieve the `hrDeviceTable` from each of the systems on your network.

Processor Table

The `hrProcessorTable` lists device-specific information about the system's processors. If the system contains only a single processor, the `hrProcessorTable` includes only one row. The columns of the `hrProcessorTable` provide the product ID of the firmware that is associated with the processor (if such an ID has been assigned and made available by the processor vendor), and the processor load. The load is the average over the last minute of the percentage of time that the processor was *not* idle.

Tracking Processor Load

By monitoring (graphing) the Processor Load average, you can visually track the load that is being placed on the processor. For example, if the graph shows constant, high levels (indicating that the processor was heavily loaded), you may decide that the host system is being overworked with too many users and processes, so you could then take steps to limit the number of users or have some of the processes execute on another system.

A temporary spike in a Processor Load graph could be an indication that a processor-intensive process is running. To avoid problems that might result from such a heavy processor load, you can reschedule that process to run at a time when processor load is less heavy.

Disk Storage Table

The hrDiskStorageTable provides information about the host's disk-storage devices. For each disk-storage device, the columns of the table list the following:

- Storage media type (for example, hard disk or floppy disk)
- Whether the disk is removable
- Storage capacity (in KB)
- Whether the disk-storage device allows read-write or read-only access

Figure 25 shows an hrDiskStorageTable for a Sun SPARC IPX.

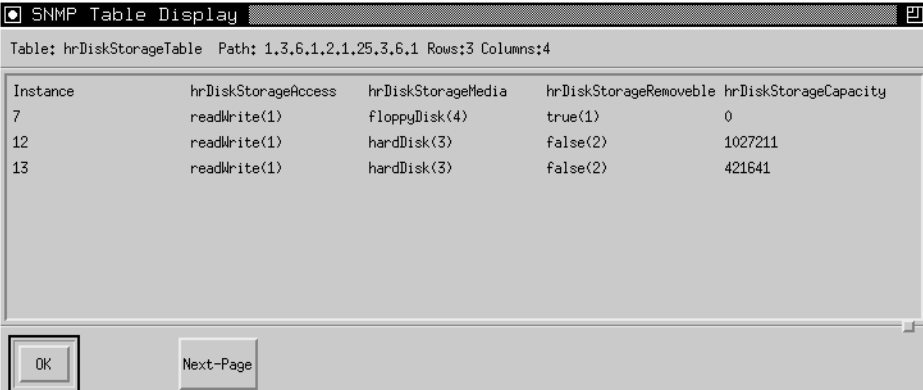


Table: hrDiskStorageTable Path: 1.3.6.1.2.1.25.3.6.1 Rows:3 Columns:4

Instance	hrDiskStorageAccess	hrDiskStorageMedia	hrDiskStorageRemovable	hrDiskStorageCapacity
7	readWrite(1)	floppyDisk(4)	true(1)	0
12	readWrite(1)	hardDisk(3)	false(2)	1027211
13	readWrite(1)	hardDisk(3)	false(2)	421641

OK Next-Page

Figure 25. Sample Host Resources Disk Storage Table

Partition Table

The hrPartitionTable shows the partitions that have been configured for the disk-storage devices. For each partition, the table lists the following:

- Textual description of the partition
- Partition ID
- Size of the partition (in KB)
- Index of the file system mounted on that partition

Figure 26 shows a sample hrPartitionTable for a Sun SPARC IPX system.

Instance	hrPartitionIndex	hrPartitionLabel	hrPartitionID	hrPartitionSize	hrPartitionFSIndex
12.1	1	Partition sda1	/dev/sda1	28905	0
12.2	2	Partition sdb1	/dev/sdb1	58515	0
12.3	3	Partition sdc1	/dev/sdc1	1023660	4
12.7	7	Partition sd1g	/dev/sd1g	936240	0
13.1	1	Partition sd0a	/dev/sd0a	16560	1
13.2	2	Partition sd0b	/dev/sd0b	65520	0
13.3	3	Partition sd0c	/dev/sd0c	414360	0
13.7	7	Partition sd0g	/dev/sd0g	178200	2
13.8	8	Partition sd0h	/dev/sd0h	154080	3

Figure 26. Sample Host Resources Partition Table

File System Table

The hrFSTable provides information about the host's file systems, both local and remote. For each file system, the table lists the following:

- Local mount point
- Remote mount point (if the file system is being mounted from a remote machine)
- Type of file system (for example, BerkeleyFFS)
- Whether the file system allows read-write or read-only access

Figure 27 shows sample output of the hrFSTable for a Sun SPARC IPX.

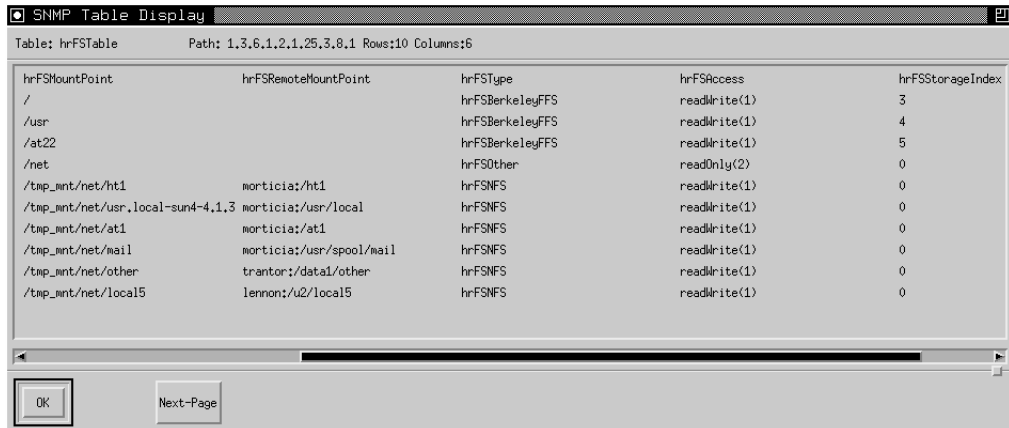


Table: hrFSSTable Path: 1.3.6.1.2.1.25.3.8.1 Rows:10 Columns:6

hrFSMountPoint	hrFSRemoteMountPoint	hrFSType	hrFSAccess	hrFSStorageIndex
/		hrFSBerkeleyFFS	readWrite(1)	3
/usr		hrFSBerkeleyFFS	readWrite(1)	4
/at22		hrFSBerkeleyFFS	readWrite(1)	5
/net		hrFSOther	readOnly(2)	0
/tmp_wnt/net/ht1	morticia:/ht1	hrFSNFS	readWrite(1)	0
/tmp_wnt/net/usr.local-sun4-4.1.3	morticia:/usr/local	hrFSNFS	readWrite(1)	0
/tmp_wnt/net/at1	morticia:/at1	hrFSNFS	readWrite(1)	0
/tmp_wnt/net/mail	morticia:/usr/spool/mail	hrFSNFS	readWrite(1)	0
/tmp_wnt/net/other	trantor:/data1/other	hrFSNFS	readWrite(1)	0
/tmp_wnt/net/local15	lennon:/u2/local15	hrFSNFS	readWrite(1)	0

Figure 27. Sample Host Resources File System Table

Host Resources Running Software Group

The hrSWRunTable lists the software that is currently running on the host system. For each running software process, the table lists the following:

- Unique identification number
- Name
- Product ID
- Directory path where the software resides
- Run-time parameters with which the software was started
- Type of software (for example, operating system or application)
- Status of the software (for example, running, runnable, not runnable, or invalid)

Figure 28 shows part of an hrSWRunTable for a Sun SPARC IPX system.

hrSWRunIndex	hrSWRunName	hrSWRunID	hrSWRunPath	hrSWRunParameters	hrSWRunType	hrSWRunStatus
1	UNIX Swapper	null			operatingSystem(2)	runnable(2)
2	init	null	/sbin/init -	/sbin/init -	application(4)	runnable(2)
3	UNIX Pager	null			operatingSystem(2)	runnable(2)
55	portmap	null	portmap	portmap	application(4)	runnable(2)
60	ypserv	null	ypserv	ypserv	application(4)	runnable(2)
62	ypxfrd	null	ypxfrd	ypxfrd	application(4)	runnable(2)
64	ypbind	null	ypbind	ypbind	application(4)	runnable(2)
66	rpc.yppupdated	null	rpc.yppupdated	rpc.yppupdated	application(4)	runnable(2)
68	keyerv	null	keyerv	keyerv	application(4)	runnable(2)
79	in.routed	null	in.routed -q	in.routed -q	application(4)	runnable(2)

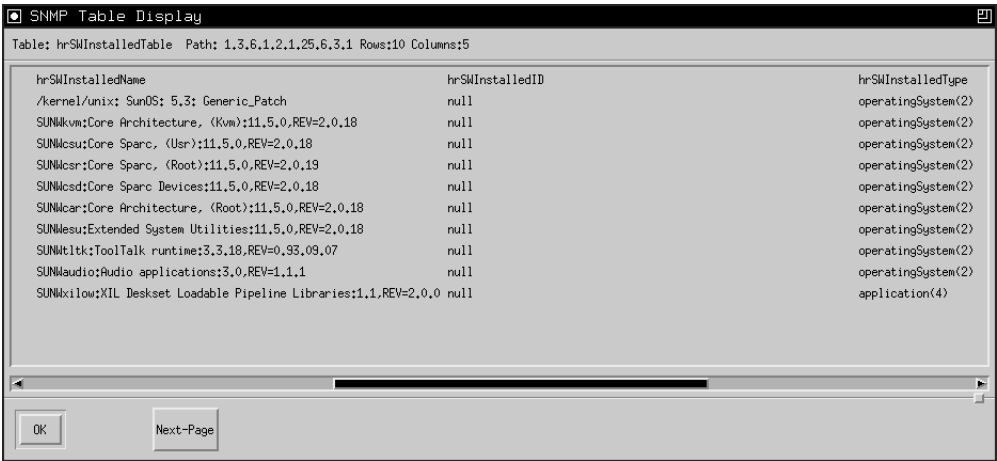
Figure 28. Sample Host Resources Running Software Table

Host Resources Installed Software Group

The hrSWInstalledTable lists the software that is currently installed on the host system. For each software package, the table lists the following:

- Unique identifier
- Software package's name
- Description
- Installation date

This table allows you to see operating system patches and versions of software that have been installed. It is ideal for checking software-version consistency between systems. Figure 29 shows part of an hrSWInstalledTable for a Solaris 2.x system.



The image shows a window titled "SNMP Table Display" with a table of installed software. The table has 10 rows and 5 columns. The columns are: hrSWInstalledName, hrSWInstalledID, hrSWInstalledType, and two unlabeled columns. The data is as follows:

hrSWInstalledName	hrSWInstalledID	hrSWInstalledType		
/kernel/unix: SunOS: 5.3: Generic_Patch	null	operatingSystem(2)		
SUNWkvm:Core Architecture, (Kvm):11.5.0.REV=2.0.18	null	operatingSystem(2)		
SUNWcsu:Core Sparc, (User):11.5.0.REV=2.0.18	null	operatingSystem(2)		
SUNWcsr:Core Sparc, (Root):11.5.0.REV=2.0.19	null	operatingSystem(2)		
SUNWcsd:Core Sparc Devices:11.5.0.REV=2.0.18	null	operatingSystem(2)		
SUNWcar:Core Architecture, (Root):11.5.0.REV=2.0.18	null	operatingSystem(2)		
SUNWesu:Extended System Utilities:11.5.0.REV=2.0.18	null	operatingSystem(2)		
SUNWltk:ToolTalk runtime:3.3.18.REV=0.93.09.07	null	operatingSystem(2)		
SUNWaudio:Audio applications:3.0.REV=1.1.1	null	operatingSystem(2)		
SUNWxilow:XIL Deskset Loadable Pipeline Libraries:1.1.REV=2.0.0	null	application(4)		

Figure 29. Sample Host Resources Installed Software Table

NOTE

Not all systems can support this table. To determine whether it is supported for the version of the operating system you are using, refer to the *eHealth SystemEDGE Release Notes*.

Unsupported MIB Objects on Windows Systems

The SystemEDGE agent for Windows supports the Host Resources MIB (RFC 1514), but some of the MIB objects within this MIB module are not supported by the underlying Windows operating system. Those objects, therefore, cannot be implemented by the Windows version of the agent.

The following Host Resources MIB objects are not supported on Windows systems:

- hrSystemInitialLoadParameters
- hrStorageAllocationFailures
- hrDeviceID (hardware manufacturers have not assigned IDs)
- hrDeviceErrors
- hrProcessorFrwID (manufacturers have not assigned IDs)
- hrPrinterTable (not implemented in current release)
- hrFSRemoteMountPoint
- hrFSLastFullBackupDate
- hrFSLastPartialBackupDate
- hrFSAccess
- hrFSBootable (not implemented in current release)
- hrSWRunEntry.SWRunPath
- hrSWRunEntry.SWRunParameters
- hrSWInstalledID

Configuring Threshold Monitoring

This chapter explains how to use the SystemEDGE agent to monitor MIB variables against user-specified thresholds.

Threshold Monitoring

The SystemEDGE agent can also monitor processes, process groups, log files, and Windows events.

The SystemEDGE agent includes a flexible Monitor table through which you can dynamically configure the agent to monitor any integer-based MIB variable under its control. You can specify the variable to monitor, the polling interval, a comparison operator (greater than, equal to, and so on), and a threshold value. The SystemEDGE agent automatically monitors that variable and sends a trap to the management system if the condition you specified is met. For more information about traps sent by the SystemEDGE agent, refer to Chapter 8, “Private Enterprise Traps.” The agent can also invoke a command on the managed system to perform management functions to correct whatever caused the exception immediately. For example, you can configure the SystemEDGE agent to monitor the percent capacity for the root file system and to notify the manager if it becomes too full.

The Monitor Table

The Monitor table, which is located in the Systems Management MIB (`empire.asn1`), provides information about each condition that the agent is currently monitoring. Each row represents a single condition that the agent is monitoring. For each entry, the table provides the following types of information:

- Variable that the agent is monitoring
- Interval at which the agent checks the variable
- Current value
- Conditions that will cause the agent to send a trap
- Number of traps that have been sent already

Sample Entry in the Monitor Table

This section provides an example of an entry in the Monitor table that instructs the SystemEDGE agent to monitor the 1-minute load average on the target system:

Index	Description	Interval	SampType	ObjID	CurrVal	Oper	Value	LastCall	#Traps	LastTrap	RowStatus	Action	Flags
12	"1min load avg"	60	absoluteValue	loadAvg1Min.0	1	gt	300	0d 22:40:00	0	0	active	""	0x0

This entry provides the following information:

- This is the twelfth row in the table.
- Its purpose is to monitor the system's 1-minute load average.
- Every 60 seconds, the agent checks the absolute value of the `loadAvg1Min` MIB variable (whose current value is 1) to see if it is greater than 300 (the specified threshold value). If it is, the agent sends a trap to any configured NMS and updates the **NumTraps** and **LastTrap** columns appropriately.

- With each poll, the agent updates the **CurrVal** column with the value it has just retrieved, records the time in the **LastCall** column, and updates the **MinValue** and **MaxValue** columns if appropriate (that is, if the value just polled is the lowest or highest value observed thus far).
- The **RowStatus** column shows that this entry is active. In this example, the action is null, so no command will be executed when the trap is sent.
- The **Flags** column is set to zero, which indicates the default Monitor table behavior. For more information about flags for the Monitor table, refer to “Monitor Table Flags” on page 252.

Columns of the Monitor Table

Table 30 defines the columns of the Monitor table. For a complete description of the Monitor table and its fields, refer to the `empire.asn1` file, located in the `/doc` subdirectory of the SystemEDGE agent distribution.

Table 30. Columns of the Monitor Table (Page 1 of 4)

Column Name	Permissions	Description	Default Value
monIndex	Read-Only	Integer (1 to MAXINT) that indicates the row index for this entry. Rows 1 through 10 are reserved for the agent’s internal use; the index for additional rows must fall in the range of 11 to MAXINT.	N/A
monDescr	Read-Write	Quoted string that is 0 to 128 characters in length and that normally contains a description of the object that is being monitored, as well as a severity level for this event.	:Default Entry:

Table 30. Columns of the Monitor Table (Page 2 of 4)

Column Name	Permissions	Description	Default Value
monInterval	Read-Write	Integer value (30 to MAXINT) that indicates how often (in seconds) the agent should monitor the variable. For example, the value 30 instructs the agent to monitor this entry every 30 seconds. This value <i>must</i> be a multiple of 30 seconds.	60
monSampleType	Read-Write	Either absoluteValue(1) or deltaValue(2); this value indicates whether this entry should sample the object's absolute value or take the difference between successive samples. For example, when monitoring counter variables, use deltaValue because it describes the rate of change. When monitoring gauges, use absoluteValue because it describes the object's exact value.	absoluteValue(1)
monOID	Read-Write	Complete object-instance identifier that represents the value to be monitored. The instance portion of the object-identifier (for example, .0 for scalars) is <i>required</i> . The object-instance must exist and must be contained within the SystemEDGE agent's supported MIBs. That is, any supported (integer-based) object that exists in MIB-II, the Host Resources MIB, or the Systems Management MIB is valid. Objects should be of integer type, including counter, gauge, integer, or enumerated integer.	0.0
monCurrVal	Read-Only	Value that was last recorded for the MIB variable that is being monitored. Every <i>monInterval</i> seconds, the agent updates this field to reflect the latest value of the variable.	N/A

Table 30. Columns of the Monitor Table (Page 3 of 4)

Column Name	Permissions	Description	Default Value
monOperator	Read-Write	<p>The operator type is a boolean operator that is used to evaluate the expression:</p> <p><i>currval operator value</i></p> <p>The operator can be one of the following:</p> <ul style="list-style-type: none"> • nop (no operation; monitor the object's value, but do not evaluate the Boolean expression) • > (greater than) • < (less than) • >= (greater than or equal to) • <= (less than or equal to) • == (equal) • != (not equal) 	nop(1)
monValue	Read-Write	Integer value to which the current value of the monitored MIB variable is compared during each monitoring cycle. If the comparison evaluates to True, the agent sends a trap. For example, if you want to be notified if the value of a gauge goes over 100, set 100 as the monValue against which the current value of the gauge is compared.	0
monLastCall	Read-Only	Time (based on sysUpTime) at which the agent last sampled (called) the MIB variable it is monitoring. 0 indicates that the MIB variable has not yet been sampled.	0
monNumTraps	Read-Only	Number of traps that have been sent for this Monitor table entry. This value provides a useful metric for determining how often the exception condition has occurred. It also provides a means to detect a missed trap message.	0

Table 30. Columns of the Monitor Table (Page 4 of 4)

Column Name	Permissions	Description	Default Value
monLastTrap	Read-Only	Time (based on sysUpTime) at which the agent last sent a trap for this Monitor table entry. 0 indicates that no traps have been sent.	0
monRowStatus	Read-Write	<p>One of the following:</p> <ul style="list-style-type: none"> • active • notInService • notReady • createAndGo • createAndWait <p>Normally, a row is either active or notInService. These values are identical in meaning to those defined by the SNMPv2 SMI RowStatus textual convention. For more information, refer to Appendix E.</p>	createAndWait(5)
monMinValue	Read-Only	Lowest (minimum) value that the agent has observed since it began polling the MIB variable.	0
monMaxValue	Read-Only	Highest (maximum) value that the agent has observed since it began polling the MIB variable.	0
monAction	Read-Write	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when the expression evaluates to True and the agent sends a trap. If the string is empty, the agent performs no action for this entry.	No action
monFlags	Read-Write	Unsigned integer flags value that indicates additional behavioral semantics that this row should follow during the course of its operation. For more information, refer to “Monitor Table Flags” on page 252.	0x0

Figure 30 shows a sample Monitor table.

monDescr	monInterval	monSampleType	monOID	monCurrVal	monOperator	monValue
1 minute load average	60	absoluteValue(1)	loadAverage1Min,0	0	gt(2)	300
5 minute load average	300	absoluteValue(1)	loadAverage5Min,0	5	gt(2)	200
15 minute load average	900	absoluteValue(1)	loadAverage15Min,0	11	gt(2)	200
Interrupt Rate	60	deltaValue(2)	numInterrupts,0	25741	gt(2)	10000
Page-fault Rate	60	deltaValue(2)	numPageFaults,0	0	gt(2)	1000
le0 Incoming Packets	60	deltaValue(2)	ifInUcastPkts,2	0	gt(2)	1000
le0 Outgoing Packets	60	deltaValue(2)	ifOutUcastPkts,2	0	gt(2)	1000
SNMP Packets	30	deltaValue(2)	snmpInPkts,0	49	gt(2)	4000
Num Processes	60	absoluteValue(1)	hrSystemProcesses,0	46	gt(2)	120
/usr filesystem	60	absoluteValue(1)	devCapacity,4	78	ge(4)	90

Figure 30. Systems Management MIB: Sample Monitor Table

Optimizing Row Creation

You can use the following MIB objects with the Monitor table to optimize row creation.

Table 31. Scalar Objects for Optimizing Row Creation

MIB Object	Permissions	Description
monUnusedIndex	Read-Only	You can use this variable to optimize table-row creation for the Monitor table. Perform an SNMP Get of this variable to return an unused index number for the Monitor table.
monMatchDescr	Read-Write	You can use this MIB object with the monMatchIndex MIB object to determine the index number that corresponds to a particular entry description. Perform an SNMP Set of this MIB object to cause the agent to search through entries in the Monitor table and place the index value of the last entry whose description matches in the monMatchIndex MIB object.
monMatchIndex	Read-Only	You can use this MIB object with monMatchDescr to match a particular entry description with its index number.

Monitor Table Flags

The monFlags column in the Monitor table is a 32-bit unsigned integer that can specify additional behavioral semantics for the corresponding Monitor table row. By default, the Monitor table row does the following:

- Attempts to reinitialize itself
- Sends SNMP traps
- Logs events through the syslog facility
- Invokes actions (if they are configured)

You can set flag bits to alter these defaults. The SystemEDGE agent interprets all flags in hexadecimal (base 16) notation. Figure 31 shows the composition of the Monitor Table flags field (monFlags).

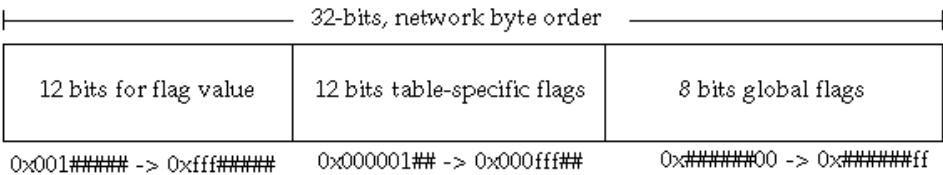


Figure 31. Monitor Table Flags

The flags value consists of three fields:

- **Field 1:** Common table flags defined for the self-monitoring tables of the Systems Management MIB. This portion is the low-order 8 bits of the flag. Figure 32 identifies and defines the individual bits in this field.
- **Field 2:** Table-specific flags that are defined separately for each of the self-monitoring tables. This field defines the next 12 low-order bits after the common table flags. Figure 33 on page 253 defines these 12 bits for the Monitor table.
- **Field 3:** Reserved 12 high-order bits for an integer value for use with table-specific flags. This field includes flags that are specific to the Monitor table.

The following sections explain each flag bit. You can combine these flag values through a logical OR operation. Figure 32 shows common flags for the monitoring tables.

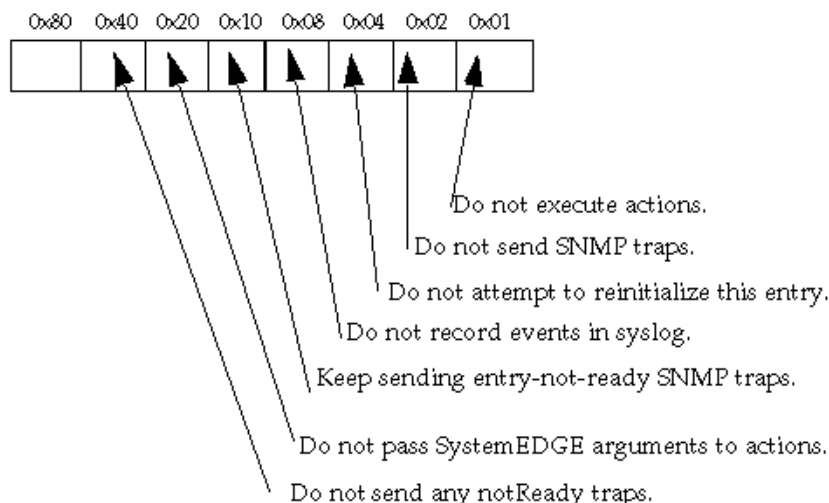


Figure 32. Common Table Flags, Low-Order 8 Bits

Figure 33 shows the flags that are specific to the Monitor table.

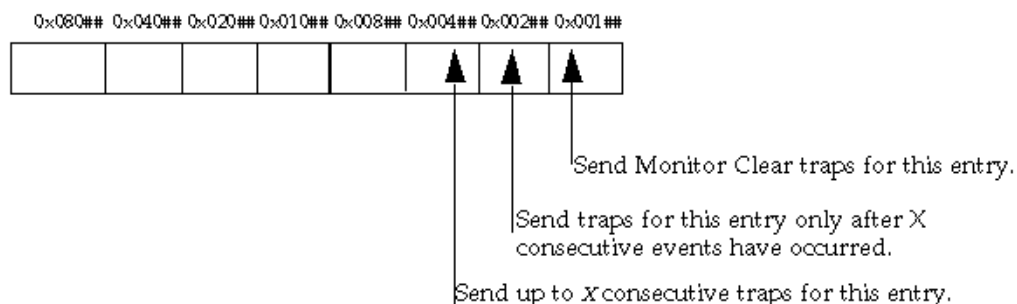


Figure 33. Monitor Table-Specific Flag Bits

Table 32 describes the Monitor Table flags.

Table 32. Monitor Table Flags (Page 1 of 2)

Flag	Description
0x00000001	Do not execute actions for this entry.
0x00000002	Do not send SNMP traps for this entry. This flag bit overrides any other flag bit with respect to traps.
0x00000004	Do not attempt to reinitialize this entry. By default, the SystemEDGE agent periodically tries to reinitialize this entry by attempting to query the MIB object it is monitoring. NOTE: Setting this bit disables automatic reinitialization.
0x00000008	Do not log events for this entry through the syslog facility. Setting this bit does not affect trap sending or threshold monitoring, but it does prevent the event from being logged through syslog. On Windows systems, the agent does not log the event to the agent's log file (sysedge.log). Disabling event logging is useful when events occur frequently or when a particular entry is used as an agent heartbeat.
0x00000010	Send continuous monitorEntryNotReady traps for this entry each time the agent attempts to reinitialize monitoring and fails to query the desired MIB object. The agent's default behavior is to send a single monitorEntryNotReady trap when a MIB object it is monitoring ceases to exist and then to attempt periodically to reinitialize the entry. Enabling this feature causes the agent to send an additional monitorEntryNotReady trap each time reinitialization fails.
0x00000020	Do not pass SystemEDGE arguments to action scripts or programs. SystemEDGE normally passes default action parameters that indicate the trap type, description field, and so on. This flag disables the passing of those arguments. For more information about action parameters, refer to "Monitor Table Actions" on page 255.
0x00000040	Do not send notReady traps for this entry.
0x00000100	Send a monitorClear trap for this entry when the threshold monitor expression transitions from True to False.

Table 32. Monitor Table Flags (Page 2 of 2)

Flag	Description
0x00000200	Send monitorEvent traps only on the <i>x</i> th consecutive event. Once the <i>x</i> th event occurs, the agent sends monitor traps for each subsequent True expression evaluation. If the threshold expression transitions from True back to False, the row resets itself, and the agent begins counting subsequent events at zero. This flag also applies to action execution. You can specify the value of <i>x</i> through the flag value field. Event logging is unaffected by this flag bit. For an example of this behavior, refer to Figure 35 on page 266.
0x00000400	Send up to <i>x</i> consecutive monitor traps, and then send no more. Enabling this feature places an upper boundary on the number of consecutive monitor traps and action executions that can occur when a threshold has been exceeded. Once the threshold expression transitions from True to False, the row resets itself, and the agent begins counting subsequent events at zero. This flag also applies to action execution. You can specify the value of <i>x</i> through the flag value field. Event logging is unaffected by this flag bit. For an example of this behavior, refer to Figure 36 on page 268.
0x###00000	Flag value 0x00100000 → 0xffff00000. Several flag bits utilize a value <i>x</i> for sending traps and executing actions. The value <i>x</i> is specified as the high-order 12 bits of the flag field. Flag bits utilizing this field are mutually exclusive. For an example of this behavior, refer to Figure 31 on page 252.

Monitor Table Actions

The SystemEDGE agent provides several default parameters to the action commands when they are invoked. These parameters are in addition to any parameters you specify in the action string and they are passed on the command line *after* those that you specify. The default action parameters are the same as the parameters that are provided in the SNMP traps that the agent sends for the Monitor table. Table 33 describes the default parameters for Monitor table actions.

Table 33. Default Parameters for Monitor Table Actions

Parameter	Description
trapType	Type of trap that is being sent. For example, monitorEvent or monitorEntryNotReadyEvent.
monDescr	Description from this table entry.
monOID	Object Identifier from this table entry.
monCurrVal	Current value from this table entry.
monValue	Value that is being monitored from this table entry.
monRowStatus	Current row status for this table entry.
monOperator	Operator that is being used from this table entry. The value passed to the action script is the numeric representation of the actual operator. For example, if the operator is >, the value passed to the action script is 2.
monIndex	Index from this table entry.
monFlags	Flags that are associated with this table entry. The value passed to the action script is in base 16 (hexadecimal) notation with a leading 0x to indicate that it is a hexadecimal number (for example, 0x00000020).

Viewing the Monitor Table with AdvantEDGE View

If you are using AdvantEDGE View, you can query a system for Monitor table information by selecting the system you want to monitor from the **System** list, selecting **Self Monitoring** from the **Configuration** list, and then clicking the **Configuration** icon. For more information, refer to the AdvantEDGE View Web Help. Figure 34 shows a sample AdvantEDGE View Monitor table.

Index	Description	Interval	Type	OID	CurrVal	Operator	Compare	Last Call	Traps	Last Trap	Min	Max	Action	Flags	Row Status
1	Monitor network interfaces, agent	30	absolute (1)	0.0	0	nop	0	49 days, 15:57:18	0	0:00:00	0	0	(no action)	0x0	●
2	CPU Load monitoring 60, agent	60	absolute (1)	0.0	0	nop	0	49 days, 15:57:18	0	0:00:00	0	0	(no action)	0x0	●
3	CPU Load monitoring 300, agent	300	absolute (1)	0.0	0	nop	0	49 days, 15:53:18	0	0:00:00	0	0	(no action)	0x0	●
4	CPU Load monitoring 900, agent	900	absolute (1)	0.0	0	nop	0	49 days, 15:43:17	0	0:00:00	0	0	(no action)	0x0	●
5	Application management module callbacks, agent	30	absolute (1)	0.0	0	nop	0	49 days, 15:57:18	0	0:00:00	0	0	(no action)	0x0	●
6	New software installation monitoring, agent	900	absolute (1)	0.0	0	nop	0	49 days, 15:43:17	0	0:00:00	0	0	(no action)	0x0	●
7	History control callback, agent	1	absolute (1)	0.0	0	nop	0	49 days, 15:57:18	0	0:00:00	0	0	(no action)	0x0	●
Add Monitor Entry															

Figure 34. Sample AdvantEDGE View Monitor Table

Assigning Entry Rows for the Monitor Table

The monIndex column of the Monitor table acts as a key field (or row index) to distinguish rows in the table. Rows 1 through 10 are reserved for internal use by the SystemEDGE agent. Users can configure rows in the range 11 to MAXINT. This section describes the benefits of reserving a block of rows for use by the system or application administrator.

Setting Local Policy

You may choose, as a matter of local policy, to reserve a block of rows for system administration. This policy allows you to define entries within a reserved block of rows without being concerned that the row might already be taken by another user's entry. In compliance with the local policy, all other users should use row indices that are outside the reserved range when they define user-configured entries.

Reserving Blocks of Rows

By reserving a block of rows, you can define a consistent set of conditions (row entries) to be monitored across all machines such that the same condition is defined in the same row number on each machine. For example, you can use row 11 (monIndex = 11) to define an entry for monitoring the swapCapacity variable, and you can then distribute this configuration to every system so that row 11 is used to monitor the swapCapacity variable on every system.

To reserve a block of rows for threshold monitoring:

1. Decide which block of rows you want to reserve for use.
2. Use that block of rows to define a set of row entries (conditions to be monitored) in the sysedge.cf initialization configuration file. For more information, refer to “Configuring the Monitor Table” on page 258.
3. Distribute the sysedge.cf file to all systems on which the SystemEDGE agent is installed.
4. Require users to avoid your block of rows when they define their own Monitor table entries.

You can also use this row-number assignment policy with AdvantEDGE View for group-configuration operations.

Configuring the Monitor Table

You can control which MIB variables and conditions the SystemEDGE agent monitors (using its threshold monitoring capability) by adding, deleting, or modifying the entries in the Monitor table.

You can configure the Monitor table in the following ways:

- **Dynamically.** Use SNMP commands from a management system, such as AdvantEDGE View, to modify the table. For more information, refer to “Dynamic Configuration During Operation” on page 261.
- **At start-up initialization.** Specify the entries for the Monitor table in the sysedge.cf file that the agent reads on start-up. For more information, refer to the next section, “Initial Configuration During Startup.”

Initial Configuration During Startup

On startup, the agent reads the `sysedge.cf` file. You can use this file to specify which MIB variables you would like SystemEDGE to monitor by adding entries with the **monitor** configuration file keyword. For more information, refer to “Adding Entries with the monitor Directive” on page 261.

Selecting Objects for Monitoring

This section describes the process of specifying MIB objects for monitoring by adding them to the `sysedge.cf` file. As an alternative, you can specify MIB objects through SNMP Set operations. For more information, refer to “Dynamic Configuration During Operation” on page 261.

To specify MIB objects for monitoring:

1. Select the SNMP object instance to be monitored. This object instance must be supported by the SystemEDGE agent and must be implemented on the platform on which the agent is running. For more information about object support, refer to the MIB specifications and the *eHealth SystemEDGE Release Notes* for your platform.

You can choose objects from the supported MIB modules: MIB-II, the Host Resources MIB, or the Systems Management MIB. (On Windows systems, you can select objects only from the Host Resources MIB and Systems Management MIB.)

NOTE

The object that you select must be of an integer-based type, such as integer, gauge, or counter. You can use textual conventions or enumerated types if they result in an integer ASN.1 value.

2. Assign the entry to a free row in the table by selecting the index number. The index number must be greater than 10, and must not yet be in use in the `sysedge.mon` file or by the agent.

3. Obtain the instance identifier for the object to be monitored.
4. Decide on the sample type. If the object you have selected is a counter, use `deltaValue`. For most other integer values (gauge, enumerated integer, integer, and so on), use `absoluteValue`.
5. Decide on the threshold and operator type against which the agent should compare the monitored variable's current value. The comparison expression that the agent uses is the following:

current-value operator value

Valid values for *operator* are described in Table 30 on page 247. Choose an appropriate value for comparison. To help select an appropriate value, you can monitor the particular object for a period of time to see what a *normal* value is. *The choice of this value is critical and depends on the semantics of the object you are monitoring.* If you want to receive `monitorClear` traps, make sure you set the appropriate bit in the `monFlags` column.

6. Choose a monitor interval in seconds. The interval *must* be a multiple of 30 seconds. Choose the interval carefully. For example, you do not want the agent to sample so frequently that an operator cannot act on the condition that is being monitored if an exception occurs.
7. Choose the monitor options, and specify the appropriate flags. For more information, refer to “Monitor Table Flags” on page 252.
8. Add the entry to the `sysedge.cf` file, and then start the agent.

Dynamic Configuration During Operation

The SystemEDGE agent uses the SNMPv2 SMI textual convention for creating, deleting, and modifying rows in the self-monitoring tables. For more information about the textual conventions for row status, refer to Appendix E.

You can dynamically modify entries in the Monitor table by sending SNMP Set request messages from your NMS (including AdvantEDGE View) to the SystemEDGE agent. Each time an SNMP request successfully modifies the Monitor table, the agent updates the `sysedge.mon` file to record the changes. This file preserves changes that are made during the operation of the SystemEDGE agent across agent and system restarts, which means that if the agent is stopped, it can restart with the same Monitor table configuration.

NOTE

Configuration file directives in `sysedge.cf` take precedence over entries in `sysedge.mon`. For example, if a Monitor table entry is contained in `sysedge.mon` at index 10, and a configuration file directive for index 10 of the Monitor table is added to `sysedge.cf`, the entry in `sysedge.cf` replaces the entry from `sysedge.mon`.

Adding Entries with the monitor Directive

Use the `monitor` keyword to add entries to the Monitor table as follows:

```
monitor attribute monIndex monFlags monInterval monSampleType monOperator  
monValue 'monDescr' 'monAction'
```

Table 34 describes the parameters for the monitor directive.

Table 34. Monitor Directive Parameters (Page 1 of 2)

Parameter	Description
<i>attribute</i>	<p>One of the following:</p> <ul style="list-style-type: none"> oid <i>variable-name</i> The word oid followed by an OID to be monitored. You can specify the OID using the complete dotted-decimal value (for example, 1.3.6.1.2.1.25.1.5.0) or the symbolic MIB name (for example, hrSystemNumUsers.0). Either way, <i>you must specify the object instance</i>, which is normally zero for non-tabular MIB variables. filesystem '<i>filesystem-name</i>' <i>variable-name</i> The word filesystem followed by the name of a mounted file system that you would like to monitor and the variable from the Systems Management MIB devTable that you would like to monitor. For instance, you can specify filesystem /usr devCapacity.
<i>monIndex</i>	Row (index) of the Monitor table to use for this entry. Each row in the agent's Monitor table is uniquely identified by an index number. Rows 1 through 10 are reserved for internal use by the agent, so the monIndex value must be greater than 10.
<i>monFlags</i>	Hexadecimal flags (for example, 0x00000001) that specify any additional behavioral semantics for this entry.
<i>monInterval</i>	<p>Integer value (30 to MAXINT) that indicates how often (in seconds) the monitoring should occur. For example, the value 30 instructs the agent to monitor this entry every 30 seconds.</p> <p>NOTE: This value must be a multiple of 30 seconds.</p>
<i>monSampleType</i>	Either absolute or delta; this value indicates whether the agent should sample the object's absolute value or take the difference between successive samples.

Table 34. Monitor Directive Parameters (Page 2 of 2)

Parameter	Description
<i>monOperator</i>	Operator type. A boolean operator that is used for evaluating the expression: <i>current-value operator value</i> The operator can be any of the following: <ul style="list-style-type: none"> • nop (no operation) • > (greater than) • < (less than) • >= (greater than or equal to) • <= (less than or equal to) • == (equal) • != (not equal)
<i>monValue</i>	Integer value (threshold) to which the current value of the monitored MIB variable is compared.
<i>monDescr</i>	Quoted string that is 0 to 128 characters in length and that normally contains a description of the object that is being monitored, as well as a severity level.
<i>monAction</i>	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when the expression evaluates to True and a trap is sent. If the string is empty, the agent performs no action for this entry.

NOTE

The SystemEDGE agent logs action-command invocations at syslog level LOG_DEBUG and action-command invocation errors at syslog level LOG_WARNING. For more information about configuring syslog, refer to Appendix B. For more information about starting the agent with its debugging options turned on, refer to “Configuring Support for Agent Debugging” on page 135.

Threshold Monitoring Examples

This section provides sample entries for the Monitor table to monitor thresholds on the target system. Each example shows how to define the entry and describes the condition that is being monitored. You can add these entries to `sysedge.cf`.

Monitoring the 1-Minute Load Average

The following examples configure the SystemEDGE agent to monitor the system's 1-minute load average:

```
monitor oid 1.3.6.1.4.1.546.1.1.7.8.26.0 11 0x00 60 absolute > 300 'Monitor 1
minute load average' ''
monitor oid loadAverage1Min.0 11 0x00 60 absolute > 300 'Monitor 1 minute load
average' ''
```

The values in these examples are defined as follows:

- 1.3.6.1.4.1.546.1.1.7.8.26.0 corresponds to the OID for the `loadAverage1Minute` variable contained within the Systems Management MIB.
- 11 indicates that this entry will occupy row 11 (`monIndex=11`) in the Monitor table.
- 60 specifies that the load average should be sampled once every 60 seconds.
- `absolute` indicates that the agent should use the object's value, not the difference between successive samples.
- 300 is the value against which the current load average is compared. If the currently sampled value is greater than (>) 300, an event occurs.

NOTE

The agent returns load averages as the underlying system's load average multiplied by 100. For example, a load average of 3 is returned as 300.

Monitoring the 5-Minute Load Average

The following example configures the SystemEDGE agent to monitor the system's 5-minute load average:

```
monitor oid loadAverage5Min.0 12 0x00500300 300 absolute > 200 'Monitor 5
minute load average' ''
```

The values in this example are defined as follows:

- loadAverage5Min.0 corresponds to the OID for the loadAverage5Minute variable contained within the Systems Management MIB.
- 12 indicates that this entry will occupy row 12 (monIndex=12) in the Monitor table.
- The flags field of 0x00500300 does the following:
 - Configures the agent to send monitorClear traps when the threshold expression transitions from True to False.
 - Configures the agent to begin sending traps (and execute actions if they are configured) only at the xth consecutive occurrence of the event.
 - Specifies that x=5. That is, the agent begins to send traps at the fifth occurrence of the event.
- 300 specifies that the load average should be sampled once every 300 seconds.
- absolute indicates that the agent should use the object's value, not the difference between successive samples.
- 200 is the value against which the current load average is compared. If the currently sampled value is greater than (>) 200, the agent sends a trap to all configured managers.
- '' indicates that no action is specified.

Figure 35 shows how the agent would send and clear traps based on this monitor directive.

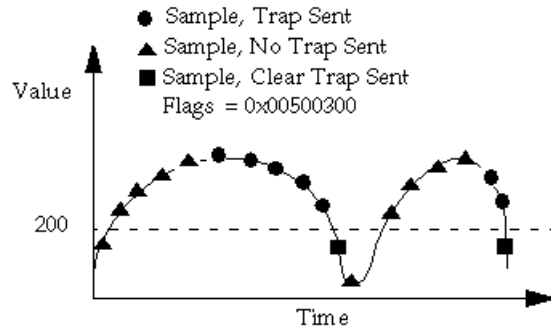


Figure 35. Monitoring the 5-Minute Load Average

Monitoring the 15-Minute Load Average

The following example configures the SystemEDGE agent to monitor the system's 15-minute load average:

```
monitor oid loadAverage15Min.0 13 0x0 900 absolute > 200 'Monitor 15 minute
load average' ''
```

The values in this example are defined as follows:

- loadAverage15Min.0 corresponds to the OID for the loadAverage15Min variable contained within the Systems Management MIB.
- 13 indicates that this entry will occupy row 13 (monIndex=13) in the Monitor table.
- 900 specifies that the load average should be sampled once every 900 seconds.
- absolute indicates that the agent should use the object's value, not the difference between successive samples.

- 200 is the value against which the current load average is compared. If the currently sampled value is greater than (>) 200, the agent sends a trap to all configured managers.
- '' indicates that no action is specified.

Monitoring the System's Interrupt Rate

The following example configures the SystemEDGE agent to monitor the rate at which hardware interrupts are occurring on the local system:

```
monitor oid numInterrupts.0 14 0x00500400 60 delta > 1000 'Monitor Interrupt  
Rate' ''
```

The values in this example are defined as follows:

- numInterrupts.0 is the OID for the numInterrupts counter object contained within the Systems Management MIB.
- 14 indicates that the entry is index 14 in the Monitor table.
- The flags field of 0x00500400 does the following:
 - Configures the agent to send a maximum of x consecutive traps when this monitor expression evaluates to True.
 - Specifies that $x=5$. That is, the agents sends 5 consecutive monitorEvent traps, and then stops sending traps until the entry resets itself. The entry resets itself when the expression transitions from True to False.
 - Does *not* specify that the agent should send monitorClear traps; consequently, a monitorClear trap is not sent when the expression transitions from True to False.
- 60 indicates that the interrupt rate should be sampled every 60 seconds.
- delta tells the agent to measure the rate at which the number of interrupts has changed. Because this object is a counter, delta is an appropriate sample type.

- 1000 is the value against which the current number of interrupts is compared.
- '' indicates that no action is specified.

Figure 36 shows how the agent would send and clear traps based on this monitor directive.

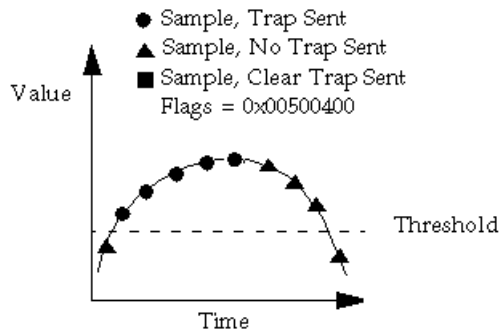


Figure 36. Monitoring the System's Interrupt Rate

Monitoring the System's Page-Fault Rate

The following example configures the SystemEDGE agent to monitor the rate at which hardware page-interrupts are occurring on the local system:

```
monitor oid numPageFaults.0 15 0x00500500 60 delta > 1000 'Monitor Page-fault
Rate' ''
```

The values in this example are defined as follows:

- numPageFaults.0 is the OID for the numPageFaults counter object contained within the Systems Management MIB.
- 15 indicates that the entry is index 15 in the Monitor table.

- The flags field of 0x00500500 does the following:
 - Configures the agent to send monitorClear Traps when the expression transitions from True to False.
 - Specifies that the agent should send at most x consecutive monitorEvent traps, and then send no more until the entry resets itself.
 - Specifies that $x=5$. That is, the agent sends 5 consecutive monitorEvent traps, and then stops sending traps until the entry resets itself.
- 60 indicates that the interrupt rate should be sampled every 60 seconds.
- delta tells the agent to measure the rate at which the number of interrupts has changed. Because this object is a counter, delta is an appropriate value for this entry's sample type.
- 1000 is the value against which the current number of interrupts is compared.
- '' indicates that no action is specified.

Figure 37 shows how the agent would send and clear traps based on this monitor directive

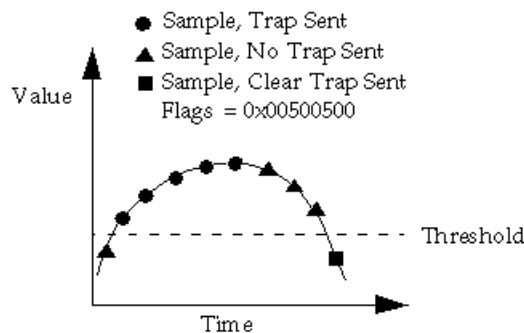


Figure 37. Monitoring the System's Page Fault Rate

Monitoring Number of Incoming Packets on the Interface

The following example configures the SystemEDGE agent to monitor the rate at which packets are received by the first ethernet interface (which is le0 for Sun systems):

```
monitor oid ifInUcastPkts.2 16 0x0 60 delta > 1000 'Monitor le0 Incoming  
Packets' ''
```

The values in this example are defined as follows:

- ifInUcastPkts.2 is the OID that indicates the particular MIB object-instance to sample.
- 16 indicates that the entry is index 16 in the Monitor Table
- 60 indicates that the agent should calculate the rate every 60 seconds.
- delta tells the agent to measure the rate at which the number of incoming packets (ifInUcastPkts) is changing.
- 1000 specifies the value to use in the comparison. If the change in rate is greater than (>) 1000, the agent sends a trap.
- '' indicates that no action is specified.

Monitoring Number of Outgoing Packets on the Interface

The following example configures the SystemEDGE agent to monitor the rate at which packets are transmitted by the first ethernet interface (which is le0 for Sun systems):

```
monitor oid ifOutUcastPkts.2 17 0x0 60 delta > 1000 'Monitor le0 Outgoing  
Packets' ''
```

The values in this example are defined as follows:

- ifOutUcastPkts.2 is the OID that indicates the particular MIB object-instance to sample.
- 17 indicates that the entry is index 17 in the Monitor table.

- 60 indicates that the agent should calculate the rate every 60 seconds.
- delta is the sample type because the object being monitored is a MIB-II ifEntry counter.
- 1000 specifies the value to use in the comparison. If the change in rate is greater than (>) 1000, the agent sends a trap message to all configured managers.
- '' indicates that no action is specified.

Monitoring Number of SNMP Packets Received

The following example configures the SystemEDGE agent to monitor the rate at which the agent receives SNMP requests:

```
monitor oid snmpInPkts.0 18 0x0 30 delta > 4000 'Monitor SNMP Packets' ''
```

The values in this example are defined as follows:

- snmpInPkts.0 is the OID that indicates the MIB II object-instance to sample.
- 18 specifies that the entry is index 18 in the Monitor table.
- 30 indicates that the agent should calculate the rate every 30 seconds.
- delta is the sample type because the object is a counter.
- 4000 specifies the value to use in the comparison. If the change in rate is greater than (>) 4000, the agent sends a trap message to all configured managers.
- '' indicates that no action is specified.

Monitoring Space on the Root File System

The following example configures the SystemEDGE agent to monitor the root (/) file system and to send a trap message when it becomes more than 95% full:

```
monitor filesystem / devCapacity 19 0x0 120 absolute > 95 'Monitor /  
Filesystem' ''
```

The values in this example are defined as follows:

- The variable identifier `devCapacity` indicates the particular MIB object-instance to monitor; in this case, the object instance is `devTableEntry.devCapacity` from the Systems Management MIB `devTable`. The object instance is not specified because it is determined automatically based on the name of the file system.
- 19 indicates that this entry is row 19 in the Monitor table.
- 120 indicates that the agent should sample every 120 seconds.
- `absolute` is the appropriate sample type because the agent is sampling an integer (not counter) value that represents how full the file system is.
- 95 specifies the value to use in the comparison. If the file system becomes greater than (>) 95% full, the agent sends a trap message to all configured managers.
- `' '` indicates that no action is specified.

Monitoring Space on the /usr File System

The following example configures the SystemEDGE agent to monitor the `/usr` file system and to send a trap message when it becomes more than 95% full:

```
monitor filesystem /usr devCapacity 20 0x00100500 120 absolute > 95 'Monitor
/usr Filesystem' ''
```

The values in this example are defined as follows:

- The variable identifier `devCapacity` indicates the particular MIB object-instance to monitor; in this case, the object instance is `devTableEntry.devCapacity` from the Systems Management MIB `devTable`. The object instance is not specified because it is determined automatically based on the name of the file system.
- 20 indicates that this entry is row 20 of the Monitor table.
- 120 indicates that the agent should sample every 120 seconds.

- absolute is the appropriate sample type because the agent is sampling an integer value that represents how full the file system is.
- 95 specifies the value to use in the comparison. If the file system becomes greater than ($>$) 95% full, the agent sends a trap message to all configured managers.
- '' indicates that no action is specified.
- The flags field of 0x00100500 specifies several things:
 - It configures the agent to send monitorClear traps when the expression transitions from True to False.
 - It specifies that the agent should send at most x consecutive monitorEvent traps and then send no more until the entry resets itself.
 - It specifies $x=1$, which indicates that the agent should send only one monitorEvent trap before waiting for the entry to reset itself.

Figure 38 shows how the agent would send and clear traps based on this monitor directive.

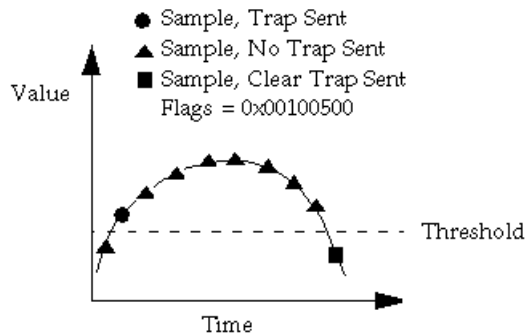


Figure 38. Monitoring Space on the /usr File System

Monitoring the Number of Processes

The following example configures the SystemEDGE agent to monitor the number of processes that are currently executing on the system and to send a trap when that number is greater than 120:

```
monitor oid hrSystemProcesses.0 21 0x0 60 absolute >= 120 'Monitor Number of  
Processes' ''
```

The values in this example are defined as follows:

- The hrSystemProcesses variable (of the Host Resources MIB) is the variable to be monitored.
- 21 indicates that this entry will be index 21 in the Monitor table.
- 60 indicates that the agent should sample the number of processes every 60 seconds.
- absolute is the sample type because the object is a gauge.
- The operator-type of greater than or equal to (\geq) instructs the agent to send a trap whenever the number of processes is greater than or equal to 120.
- '' indicates that no action is specified.

Using the edgemon Utility to Monitor Thresholds

edgemon is a command-line utility that automatically configures the SystemEDGE agent to monitor a MIB variable that you specify. With this utility, you specify the following:

- MIB variable, either by name or by object-identifier
- Threshold value and comparison operator
- Flags
- Description
- Optional action

The edgemon utility then issues an SNMP Set request to create the appropriate entry in the target agent's Monitor table.

Use the edgemon utility as follows:

```
edgemon ipaddr[:port][,timeout] commstr [command]
```

Table 35 describes the parameters to the edgemon utility.

Table 35. edgemon Parameters

Parameter	Description
<i>ipaddr[:port][,timeout]</i>	Specifies the hostname or IP address of the system on which the agent exists. If the agent is running on an alternative UDP port (for example, 1691), you must specify that port number and the hostname or IP address through a colon-separator. In addition, you can specify an optional SNMP timeout value (in seconds) using a command-separator.
<i>commstr</i>	Specifies the community string that edgemon uses in its SNMP requests to the agent. Because edgemon uses SNMP Set requests, use a community string that provides read-write access to the target agent.
<i>command</i>	<p>Specifies the command and associated arguments. Supported commands include the following:</p> <ul style="list-style-type: none"> • oid (for monitoring an object) • filesystem (for monitoring a file system) • list (for listing the current Monitor table entries) • setstatus (for setting the status of an Monitor table entry) • delete (for deleting a Monitor table entry) <p>For more information about these commands, refer to the next section, “edgemon Commands for Threshold Monitoring.”</p>

edgemon Commands for Threshold Monitoring

The edgemon command and associated arguments are as follows:

```
oid object-instance index flags interval stype oper val "descr" "action"
filesystem filesystem-name variable-name index flags interval stype oper val
"descr" "action"
list
setstatus index status
delete index
```

Table 36 defines the parameters for the edgemon command.

Table 36. edgemon Command Parameters (Page 1 of 2)

Parameter	Description
<i>object-instance</i>	Specifies the object-identifier or object-name to monitor. You can specify the OID using the complete dotted-decimal value (for example, 1.3.6.1.2.1.25.1.5.0) or the symbolic MIB name (for example, hrSystemNumUsers.0). Either way, you must specify the instance, which is normally 0 for non-tabular MIB variables. The object-instance or object-name must point to a MIB object that exists within the Systems Management MIB, the Host Resources MIB, or MIB-II. On Windows systems, this object instance or object name can only point to an object within the Systems Management MIB or Host Resources MIB.
<i>index</i>	Specifies the row (index) of the Monitor table to use for this monitoring entry. Each row in the agent's Monitor table is uniquely identified by an index number. Rows 1 through 10 are reserved for internal use by the agent, so the monIndex value must be greater than 10.
<i>flags</i>	Specifies the hexadecimal flags (for example, 0x00000001) that direct the additional behavioral semantics of this Monitor table entry. For a list of flags, refer to Table 34 on page 262.
<i>interval</i>	Specifies an integer value (30 to MAXINT) that indicates how often (in seconds) the monitoring should be performed. For example, the value 30 instructs the agent to monitor this entry every 30 seconds. NOTE: This value must be a multiple of 30 seconds.

Table 36. edgemon Command Parameters (Page 2 of 2)

Parameter	Description
<i>stype</i>	Specifies sample type of either absolute or delta. This value indicates whether the agent should sample the object's absolute value or take the difference between successive samples.
<i>oper</i>	Specifies the boolean operator to use when comparing the sampled value to the threshold value. The operator can be one of the following: <ul style="list-style-type: none"> • nop • > (greater than) • < (less than) • >= (greater than or equal to) • <= (less than or equal to) • == (equal) • != (not equal)
<i>val</i>	Specifies the integer value (threshold) to which the current value of the monitored MIB variable is compared.
<i>"descr"</i>	Specifies a quoted string that is 0 to 128 characters in length and that normally contains a description of the object that is being monitored, as well as a severity level.
<i>"action"</i>	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when the expression evaluates to True and a trap is sent. If the string is empty, no action is performed for this entry.
<i>filesystem-name</i>	Specifies the name of the mounted file system to monitor.
<i>variable-name</i>	Specifies the file system variable from the Systems Management MIB devTable to monitor. For example, you can monitor a file system's capacity by specifying devCapacity as the variable-name.
<i>status</i>	One of the following: <ul style="list-style-type: none"> • active (activate a row) • notInService (deactivate but preserve a row) • destroy (delete a Monitor table row)

edgemon Examples

This section provides examples of how to use the edgemon command.

Monitoring 1-Minute Load Average with edgemon

The following example creates an entry at index 11 in the agent's Monitor table that monitors the system's 1-minute load average for a threshold of 3:

```
edgemon 143.45.0.12 private oid loadAverage1Min.0 11 0x00 60 absolute > 300  
"Monitor 1 minute load average" ""
```

Monitoring Hardware Interrupts with edgemon

The following example creates a Monitor table entry to monitor the number of hardware interrupts on the underlying system against the threshold 1000:

```
edgemon 143.45.0.12 private oid numInterrupts.0 14 0x00500400 60 delta > 1000  
"Monitor Interrupt Rate" ""
```

The agent creates this entry at index 14 and samples the numInterrupts variable every 60 seconds. The flags field of 0x00500400 instructs the agent to modify the default Monitor table behavior as follows:

- **0x00000400.** Instructs the agent to send up to *x* consecutive monitorEvent traps and then send no more.
- **0x00500000.** Contains the flag value *x*=5 for use with the directive above.

Monitoring the /usr File System with edgemon

The following example creates a Monitor table entry at index 20 to monitor the system's /usr file system for a capacity that is greater than or equal to 95%, checking the file system every 2 minutes (120 seconds):

```
edgemon 143.45.0.12 private filesystem /usr devCapacity 20 0x00100500 120
absolute >= 95 "Monitor /usr Filesystem" ""
```

Removing Threshold Monitoring Entries

To stop the threshold monitoring of a MIB variable, you must remove the appropriate entry from the Monitor table. This requires that you remove the entry from the both Monitor table and from the sysedge.cf file.

As described in “Dynamic Configuration During Operation” on page 261, Monitor table entries are stored in the file sysedge.mon to ensure that they are not lost when the SystemEDGE agent is restarted. The monitor directives in the sysedge.cf file create a monitor entry in sysedge.mon whenever the SystemEDGE agent is started.

Removing Entries from the sysedge.cf File

If you configured a Monitor table entry by adding a monitor directive to the sysedge.cf file, you must delete it manually from sysedge.cf. If you do not remove the sysedge.cf directive, the entry will be recreated in sysedge.mon the next time the SystemEDGE agent is restarted.

Removing Entries with the edgemon Utility

To remove a threshold-monitoring entry from the Monitor table, use the edgemon utility to delete the entry. The following example deletes row 14 from the Monitor table on host 143.45.0.12. Once deleted, the row is removed from memory and from the sysedge.mon file.

```
edgemon 143.45.0.12 private delete 14
```

Removing Entries Manually

In some cases, you may be unable to use the edgemon utility to delete Monitor table entries. For example, if you have configured the SystemEDGE agent to disallow SNMP Set operations, the edgemon utility does not work. In this case, you need to remove the threshold-monitoring entry from the Monitor table by editing the `sysedge.mon` file. Because this is an active file, you need to stop the SystemEDGE agent before you edit the file. For more information on the format of the `sysedge.mon` file, refer to Appendix C.

For example, to delete row 14, do the following:

1. Stop the SystemEDGE agent.
2. Open `sysedge.mon` for editing, delete the entry for monentry row 14, and save the file.
3. Open `sysedge.cf` for editing, delete the entry for monentry row 14 if it exists, and save the file.
4. Restart the SystemEDGE agent.

Configuring Process and Service Monitoring

This chapter explains how to use the SystemEDGE agent to monitor processes and services. The SystemEDGE agent can monitor many attributes of a process, including whether it is running, its size, CPU and memory and usage, and the number of disk and network I/O operations.

Monitoring Processes and Windows Services

For information about monitoring process groups, refer to Chapter 12, “Configuring Process Group Monitoring.”

The flexible Process Monitor table of the Systems Management MIB enables you to configure the agent dynamically to monitor specific attributes of important processes, services, and applications that are running on the underlying system. You specify the process, attribute, threshold value, and interval that you want to monitor. If a process attribute crosses the threshold you specified, the agent sends an SNMP trap to the management systems you have configured. The agent can also invoke an action command on the managed system to immediately correct the problem.

Monitoring Windows Services

On UNIX systems, services or daemons are processes, which can be monitored just like any other process. On Windows systems, however, services are special kinds of processes that are started and stopped using a graphical interface (for example, through the Services Control Panel). Windows services run

within processes, but the mapping between them is not always one to one. For example, multiple Windows services may run *within* a single process. Consequently, you can monitor Windows services in two ways:

- Monitor Windows services by monitoring their underlying processes. It is not always easy to figure out the underlying process within which a service is running. In this case, you can have the SystemEDGE agent monitor the service itself rather than the underlying process.
- Instruct the SystemEDGE agent to monitor the Windows service, rather than the underlying process, by setting a flag or by using the configuration keyword `watch ntsservice`. For more information about Process Monitor table flags, refer to Table 40 on page 291.

NOTE

Because Windows does not track process attributes for Windows services, the SystemEDGE agent can monitor *only* the `procAlive` attribute for Windows services. To monitor other attributes (for example, `procRSS`) for a particular Windows service, you must monitor the underlying process that implements that Windows service.

Sample Process Monitor Table Entry

This section provides a sample Process Monitor table entry that instructs the SystemEDGE agent to monitor the `httpd` process to make sure it is up and running.

Index	Description	Interval	SampType	Attribute	CurrVal	Oper	Value	LastCall	#Traps	LastTrap	Flags	RowStatus	Action	RegExpr
12	"Monitor httpd"	60	absoluteValue	procAlive	3	gt	4	0d 22:40:00	0	0	0x00000100	active	''	httpd

This sample entry provides the following information:

- This entry is the 12th row in the Process Monitor table.
- It instructs the SystemEDGE agent to monitor the Web server daemon (`httpd`) every 60 seconds.

- The attribute being monitored is the status of the process (procAlive).
- The current value of 3 indicates the process is running normally (as far as the operating system is concerned). If the process status goes to 4, or if the process stops running, the agent will send a processStop trap to the management system.
- The **RowStatus** column shows that this entry is active.
- The **Action** column is null (' '), which indicates that the agent invokes no action when the process stops running.
- The **Flags** value (0x00000100) instructs the agent to monitor the parent httpd daemon, rather than child processes. For more information on Process Monitor table flags, refer to “Process Monitor Table Flags” on page 289.

The Process Monitor Table

The Process Monitor table provides information about each of the process/attribute pairs (or Windows service-status pairs) that the agent is currently monitoring. Each row in the table represents the combination of a process or Windows service and a particular attribute of that process or service that the agent is monitoring.

For each entry, the table provides the following types of information:

- Attribute that is being monitored
- Interval at which the agent checks the attribute
- Process that the agent is watching
- Number of traps that have been sent
- Current attribute value

Columns of the Process Monitor Table

Table 37 describes the columns of the Process Monitor table. For a complete description of the Process Monitor Table, refer to the Systems Management MIB specification (empire.asn1 in the /doc subdirectory of the SystemEDGE agent distribution).

Table 37. Columns of the Process Monitor Table (Page 1 of 4)

Column Name	Permissions	Description
pmonIndex	Read-Only	Integer (1 to MAXINT) that indicates the row index for this entry.
pmonDescr	Read-Write	Quoted string that is 0 to 128 characters in length and that normally contains a description of the process and attribute that the agent is monitoring, as well as a severity level.
pmonInterval	Read-Write	Integer value (30 to MAXINT) that indicates how often (in seconds) the agent should perform this monitoring. For example, the value 30 instructs the agent to monitor this entry every 30 seconds. NOTE: This value must be a multiple of 30 seconds.
pmonSampleType	Read-Write	Either absoluteValue(1) or deltaValue(2); this value indicates whether the agent should sample the attribute's absolute value or take the difference between successive samples. For example, use deltaValue to monitor counter attributes because it provides the rate of change. Use absoluteValue to monitor gauges, because it provides the object's exact value.
pmonAttribute	Read-Write	Process attribute that is being monitored. For a complete list of supported attributes, refer to Table 38 on page 288. For example, to monitor a process to ensure that it is alive, specify the procAlive attribute. To track the number of packets received by the particular application or process, specify procMsgsSent.

Table 37. Columns of the Process Monitor Table (Page 2 of 4)

Column Name	Permissions	Description
pmonCurrVal	Read-Only	<p>Attribute value that was last recorded for the process that is being monitored. Every <i>pmonInterval</i> seconds, the agent updates this field to reflect the latest reading for the attribute.</p> <p>When monitoring <i>procAlive</i>, this value is mapped from the <i>hrSWRunStatus</i> variable of the Host Resources MIB. Possible values follow:</p> <ul style="list-style-type: none"> • 0 – The row is not ready. (If this is the case, the PID value will be -1.) • 1 – The process is running. • 2 – The process is waiting for a resource (CPU, memory, or I/O). • 3 – The process is not runnable. It is waiting for an event. • 4 – Invalid. The process is not loaded.
pmonOperator	Read-Write	<p>Operator type. A boolean operator used for evaluating the following expression:</p> <p><i>current-value operator value</i></p> <p>The operator can be one of the following:</p> <ul style="list-style-type: none"> • nop (no operation; monitor the object's value, but do not evaluate the Boolean expression) • > (greater than) • < (less than) • >= (greater than or equal to) • <= (less than or equal to) • == (equal) • != (not equal)
pmonValue	Read-Write	<p>Integer value to which the current value of the monitored process attribute is compared during each monitoring cycle. If the comparison evaluates to True, the agent sends a trap. For example, if you want to be notified if the value of a gauge goes above 100, set 100 as the <i>pmonValue</i> to which the agent compares the current value of the gauge.</p>

Table 37. Columns of the Process Monitor Table (Page 3 of 4)

Column Name	Permissions	Description
pmonLastCall	Read-Only	Time (based on sysUpTime) at which the agent last sampled (called) the process attribute it is monitoring. 0 indicates that the MIB variable has not yet been sampled.
pmonNumTraps	Read-Only	Number of traps that have been sent for this entry. This column provides a useful metric for determining the frequency at which the exception condition is occurring and a means for detecting missed trap messages.
pmonLastTrap	Read-Only	Time (based on sysUpTime) at which the agent last sent a trap for this entry. 0 indicates that no traps have been sent.
pmonFlags	Read-Write	Integer flags value that indicates additional behavioral semantics this row should follow during the course of its operation. The default is 0x00. For more information about this field, refer to “Process Monitor Table Flags” on page 289.
pmonAction	Read-Write	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when the expression evaluates to True and the agent sends a trap. If the string is empty, no action will be performed for this entry. By default, no action is performed.
pmonRegExpr	Read-Write	Specifies the regular expression to apply when the agent is attempting to acquire the process ID of an application or a process to monitor. For Windows service monitoring, this regular expression is used to match the name of the Windows service to monitor. Rather than requiring users to specify process IDs (PIDs) or service indexes, which may change, users specify a regular expression for process name or service name. The agent uses this user-specified name to find the process to monitor. By default, the Process Monitor table keeps attempting to apply the regular expression until a new PID or service is found if the process or service stops running.
pmonMinValue	Read-Only	Lowest (minimum) value that the agent has observed since it began polling the process attribute. The default is 0.

Table 37. Columns of the Process Monitor Table (Page 4 of 4)

Column Name	Permissions	Description
pmonMaxValue	Read-Only	Highest (maximum) value that the agent has observed since it began polling the process attribute. The default is 0.
pmonCurrentPID	Read-Write	PID of the process/attribute pair that is currently being monitored.
pmonRowStatus	Read-Write	<p>One of the following:</p> <ul style="list-style-type: none"> • active • notInService • notReady • createAndGo • createAndWait <p>Normally, a row is either active or notInService. These values are defined in the SNMPv2 SMI RowStatus textual convention. For more information, refer to Appendix E.</p>

Process Attributes

Table 38 describes the attributes that the SystemEDGE agent can monitor for a particular process or service. The table also specifies the SNMP type for each attribute. You can use the SNMP type to select the sample type and operator. For example, absoluteValue sampling is usually most appropriate for attributes of type integer or gauge, while deltaValue sampling is usually most appropriate for attributes of type counter.

NOTE

The agent's ability to monitor a particular process attribute is dependent on the underlying operating system's ability to track the associated parameter or metric.

Table 38. Process Attributes

Process Attribute	SNMP Type	Description
procAlive	Boolean	Specifies whether the process or service is running
procMEM	Gauge	Percentage (0 to 100) of real memory used by this process
procSize	Gauge	Size of text, data, and stack segments (KB)
procRSS	Gauge	Real memory (resident set) size of the process (KB)
procTime	Integer	Accumulated CPU time in seconds for this process
procInBlks	Counter	Number of blocks of data input by the process
procOutBlks	Counter	Number of blocks of data output by this process
procMsgsSent	Counter	Number of messages received by this process
procMsgsRecv	Counter	Number of messages sent by this process
procNice	Integer	Priority (nice value) of this process
procNumThreads	Integer	Number of threads that are running within this process
procNumSwaps	Counter	Number of times this process has been swapped
procSysCalls	Counter	Number of system calls invoked by this process
procMinorPgFlts	Counter	Number of minor page faults incurred by this process
procMajorPgFlts	Counter	Number of major page faults incurred by this process
procVolCtx	Counter	Number of voluntary context switches incurred by this process
procInvolCtx	Counter	Number of involuntary context switches incurred by this process

Optimizing Row Creation

You can use the following MIB objects with the Process Monitor table to optimize row creation.

Table 39. Scalar Objects for Optimizing Row Creation

MIB Object	Permissions	Description
pmonUnusedIndex	Read-Only	You can use this variable to optimize table-row creation for the Process Monitor table. Perform an SNMP Get of this variable to return an unused index number for the Process Monitor table.
pmonMatchDescr	Read-Write	You can use this MIB object with the pmonMatchIndex MIB object to determine the index number that corresponds to a particular entry description. Perform an SNMP Set of this MIB object to cause the agent to search through entries in the Process Monitor table and place the index value of the last entry whose description matches in the pmonMatchIndex MIB object.
pmonMatchIndex	Read-Only	You can use this MIB object with pmonMatchDescr to match a particular entry description with its index number.

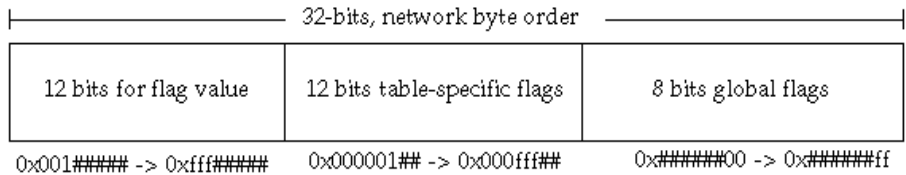
Process Monitor Table Flags

The **pmonFlags** column in the Process Monitor table is a 32-bit unsigned integer field that can specify additional behavioral semantics for the corresponding row.

By default, the Process Monitor table row does the following:

- Attempts to reinitialize itself
- Sends SNMP traps
- Logs syslog events
- Invokes actions (if they are configured)

You can set different flag bits to alter these defaults. The agent interprets all flags in hexadecimal (base 16) notation. Figure 39 shows the composition of the Process Monitor table flags (pmonFlags) field.

**Figure 39. Process Monitor Table Flags**

The flags value consists of three fields:

- **Field 1:** Common table flags for the self-monitoring tables of the Systems Management MIB. This portion is the low-order 8 bits of the flags field.
- **Field 2:** Table-specific flags that are defined separately for each of the self-monitoring tables. This field defines the next 12 low-order bits after the common table flags. Figure 40 on page 291 shows how these 12 bits are defined for the Process Monitor table.
- **Field 3:** Reserved 12 high-order bits for an integer value for use in conjunction with table-specific flags. This field defines the flags that are specific to the Process Monitor table.

The following sections explain each flag bit. You can combine flag values through a logical OR operation. Figure 40 shows the flags that are specific to the Process Monitor table.

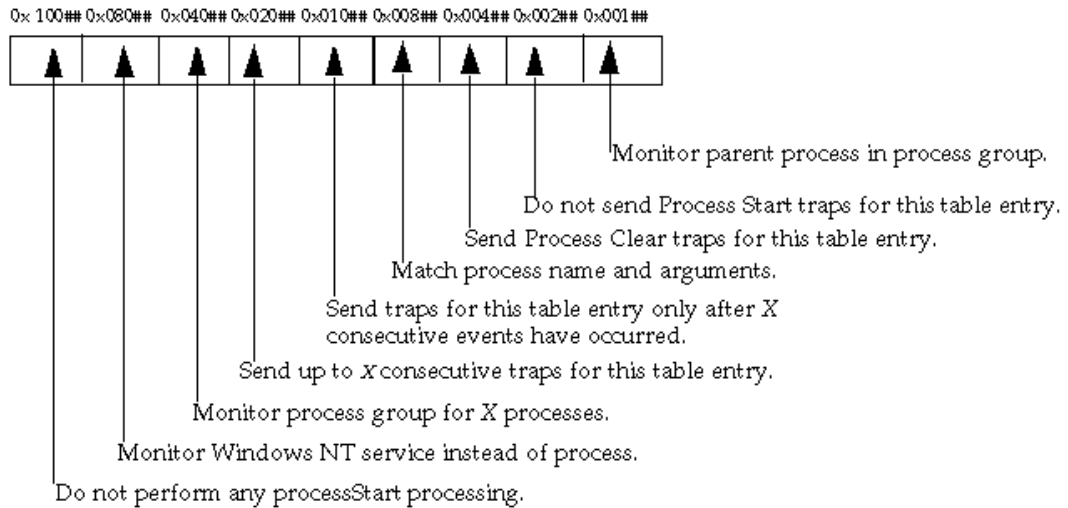
**Figure 40. Process Monitor Table-Specific Flags**

Table 40 describes the Process Monitor table flags.

Table 40. Process Monitor Table Flags (Page 1 of 4)

Flag	Description
0x00000001	Do not execute actions for this entry.
0x00000002	Do not send SNMP traps for this entry. This flag bit overrides any other flag bit with respect to traps.
0x00000004	Do not attempt to reinitialize this entry. By default, the agent periodically tries to reinitialize this entry by scanning the process table to determine the new process ID if the target process has been restarted. Setting this bit disables automatic reinitialization.
0x00000008	Do not record events for this entry in the syslog facility. Setting this bit does not affect trap sending or threshold monitoring. It prevents the event from being logged through syslog only. On Windows systems, it prevents logging events to the agent's log file (sysedge.log). When events occur frequently, it may be useful to disable event logging.

Table 40. Process Monitor Table Flags (Page 2 of 4)

Flag	Description
0x00000010	<p>Send continuous processStop traps for this entry each time the agent attempts to reinitialize process monitoring and fails to match a process. The agent's default behavior is to send a single processStop trap when a process dies and then attempt to periodically reinitialize the entry. Enabling this feature causes the agent to send an additional processStop trap each time reinitialization fails. In all cases, the agent does not send processStart and processStop traps unless the corresponding entry is monitoring the procAlive process attribute.</p> <p>NOTE: This flag is valid only when the agent is monitoring the procAlive attribute. When you are monitoring Windows services, an entry does not enter the notReady state when the service is not running. Setting this flag causes SystemEDGE to generate processStop traps—and execute any associated actions—for the entry when the service is not running, even though it remains in the ready state.</p>
0x00000020	<p>Do not pass SystemEDGE arguments to action scripts or programs. SystemEDGE normally passes default action parameters that indicate the trap type, description field, and so on. This flag disables the passing of those arguments. For more information, refer to “Process Monitor Table Actions” on page 296.</p>
0x00000040	<p>Do not send notReady traps for this entry.</p>
0x00000100	<p>Monitor the parent process in the process group. Many applications and services (for example, Web server httpd daemons) are designed such that an initial daemon spawns child processes to handle actual service requests. These child processes often service several requests and then exit. In these cases, it is preferable to monitor the main parent daemon rather than the child processes. Enabling this feature causes the agent to search for and monitor the parent daemon rather than the first process it finds that matches the process regular expression.</p> <p>The agent performs this search as follows: It scans the Process Monitor table for processes that match the name of the process regular expression (pmonRegExpr). If the matching parent process of the process also matches, the agent returns the parent process. This searching algorithm only accommodates parent/child relationships and cannot handle daemons forking daemons. This feature is not available on Windows systems because the notion of parent/child processes does not exist on Windows.</p> <p>NOTE: You <i>cannot</i> set this flag if you are using the 0x00000800 flag.</p>

Table 40. Process Monitor Table Flags (Page 3 of 4)

Flag	Description
0x00000200	Do not send processStart traps for this entry. If this feature is enabled, the agent sends processStop traps and logs events (unless those features were disabled through their corresponding flags bits).
0x00000400	Send processClear traps for this entry when a process monitor expression transitions from True to False. This feature is only applicable when the attribute being monitored is not procAlive. Figure 41 on page 294 shows how to use this option to send a processClear trap. Figure 42 on page 295 shows the default behavior, whereby processClear traps are not sent.
0x00000800	Match process name and arguments when targeting a process for monitoring. By default, the agent matches only against a process name. Enabling this option causes the agent to apply the pmonRegExpr to both the process name and process arguments, which is sometimes necessary to distinguish between similar processes or multiple invocations of the same application or binary. NOTE: This flag is valid for UNIX systems only. You <i>cannot</i> set this flag if you are using the 0x00000100 flag.
0x00001000	Send processThreshold traps only after the xth consecutive event. Enabling this feature instructs the agent to wait until the xth consecutive occurrence of an event before sending processThreshold traps. Once the xth event has occurred, the agent will send processThreshold traps for each subsequent, consecutive True expression evaluation. If the threshold expression transitions from True back to False, the row resets itself, and the agent begins counting events from zero. This flag also applies to action execution. You can specify the value of x is specified through the flag value field. Event logging is unaffected by this flag bit. For an example, refer to Figure 43 on page 295.
0x10000	Do not perform any processStart processing. If this flag is enabled, SystemEDGE does not invoke actions, log events, or send traps when processStart events occur.
0x00002000	Send up to x consecutive processThreshold traps, and then send no more. Enabling this feature places an upper boundary on the number of consecutive processThreshold traps and action executions that can occur when a process has exceeded a threshold. Once the threshold expression transitions from True to False, the row resets itself, and the agent begins counting events from zero. This flag also applies to action execution. You can specify the value of x through the flag value field. Event logging is unaffected by this flag bit. For an example, refer to Figure 44 on page 296.

Table 40. Process Monitor Table Flags (Page 4 of 4)

Flag	Description
0x00004000	Monitor a process group for <i>x</i> processes.
0x00008000	Monitor the Windows service that matches the corresponding regular expression. Setting this flag instructs SystemEDGE to monitor the procAlive attribute of the matching Windows service within the Windows service table. It is not necessary to set this flag bit if you use the watch ntservice configuration file directive.
0x###00000	Flag value 0x00100000 → 0xffff00000. Several flag bits utilize a value <i>x</i> for sending traps and executing actions. The value <i>x</i> is specified as the high-order 12 bits of the flag field. Flag bits utilizing this field are mutually exclusive. For more information, refer to Figure 39 on page 290.

Figure 41 shows the agent sending one trap to indicate that the monitored object crossed the threshold, and then sending a processClear trap when value drops below the threshold again.

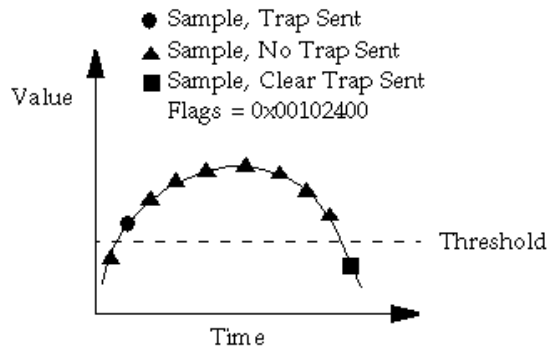
**Figure 41. Sending a processClear Trap**

Figure 42 shows the SystemEDGE agent sending four traps to indicate that the value of the monitored object is above the threshold. It does not send a processClear trap when the value falls below the threshold again.

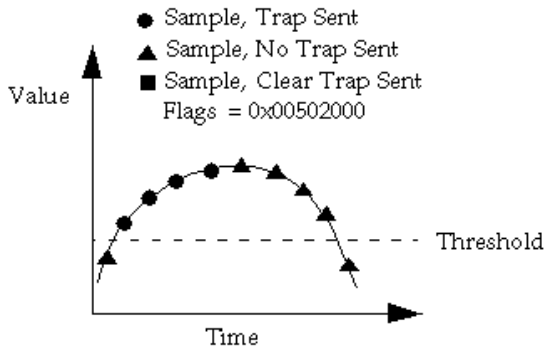


Figure 42. Sending a Limited Number of Traps with No processClear Trap

Figure 43 shows the agent waiting until an event has occurred several times before it begins sending traps. It then sends traps until the value of the monitored object falls below the threshold, at which time it sends a processClear trap.

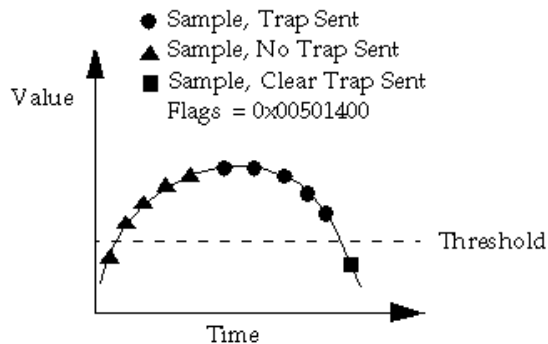


Figure 43. Sending Traps Only After Multiple Occurrences of an Event

Figure 44 shows the agent sending traps only a specified number of times. When the value of the monitored object falls below the threshold, the agent sends a processClear trap.

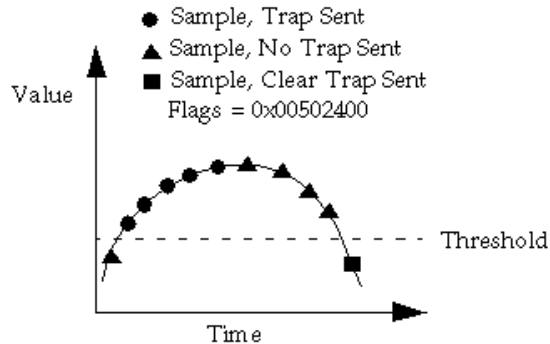


Figure 44. Limiting the Number of Traps Sent

Process Monitor Table Actions

The SystemEDGE agent provides several default parameters to the action commands when they are invoked. These parameters are in addition to any parameters that you specify in the action string and are passed on the command line after those that you specify. The default parameters are the same as the parameters provided in the SNMP traps that are sent for the Monitor table. Table 41 describes the default parameters for Process Monitor table actions.

Table 41. Default Parameters for Process Monitor Table Actions

Parameter	Description
trapType	Type of trap that is being sent: <ul style="list-style-type: none"> • processStop • processStart • processThreshold • processClear
pmonIndex	Index that is assigned to this table entry.
pmonDescr	Table entry's description.
pmonAttribute	Process attribute that is being monitored by this entry.
pmonCurrVal	Current value obtained for this table entry.
pmonOperator	Operator that is being used by this table entry.
pmonValue	Comparison value or threshold that is applied by this table entry.
pmonFlags	Flags that are associated with this table entry.
pmonRegExpr	Regular expression that the SystemEDGE agent uses to find the process ID of the process to monitor.
pmonCurrentPID	Process ID (PID) of the current process being monitored. If you are monitoring a Windows service (through the watch ntservice directive or by setting flag 0x8000), this column represents the index in the NT Service MIB table that this process or service monitoring entry has acquired.

For more information about traps sent by the SystemEDGE agent, refer to Chapter 8, "Private Enterprise Traps."

Viewing the Process Monitor Table with AdvantEDGE View

If you are using AdvantEDGE View, you can query a system for Process Monitor table information by selecting the system you want to monitor from the **System** list, selecting **Process Monitoring** from the **Configuration** list, and clicking the **Configuration** icon. For more information, refer to the AdvantEDGE View Web Help. Figure 45 shows a sample AdvantEDGE View Process Monitor table.




Index	Description	Interval	Type	Attribute	Curr Val	Oper	Compare	Last Call	Traps	Last Trap	Flags	Action	RegExpr	Min	Max	PID	Row Status
503	Monitor ypbind (a group entry)	60	absolute	procAlive (1)	0	>	invalid(4)	49 days, 16:53:20	0	0:00:00	0x1	(no action)	ypbind	0	0	-1	
1000	Monitor inetd	60	absolute	procAlive (1)	notRunnable (3)	>	invalid(4)	49 days, 16:53:20	0	0:00:00	0x100	(no action)	inetd	3	3	150	
5000	Group monitor inetd	30	absolute	procAlive (1)	notRunnable (3)	==	invalid(4)	49 days, 16:53:50	0	0:00:00	0x100	(no action)	inetd	3	3	150	
Add Process Monitor Entry																	

Figure 45. Sample AdvantEDGE View Process Monitor Table

Assigning Entry Rows for the Process Monitor Table

The pmonIndex column of the Process Monitor table acts as a row index to distinguish rows in the table. Rows 1 through 10 are reserved for internal use by the SystemEDGE agent. Users can configure rows in the range of 11 to MAXINT. For more information about reserving blocks of rows, refer to “Reserving Blocks of Rows” on page 258.

Configuring the Process Monitor Table

You can control the processes and process attributes that the SystemEDGE agent monitors by adding, deleting, or modifying the entries in the Process Monitor table.

You can configure the Process Monitor table in the following ways:

- **Dynamically.** Use SNMP commands from a management system, such as AdvantEDGE View, to modify the table. For more information, refer to the next section, “Dynamic Configuration During Operation.”
- **At start-up initialization.** Specify the process attributes to monitor through the agent’s configuration file `sysedge.cf`. For more information, refer to “Initial Configuration During Startup” on page 300.

You can also dynamically add, delete, or modify Process Monitor Table entries through the `edgwatch` utility. For more information, refer to “Using the `edgwatch` Utility to Monitor Processes” on page 310.

Dynamic Configuration During Operation

The SystemEDGE agent uses the SNMPv2 SMI Row Status textual convention for creating, deleting, and modifying rows in the table. For more information about the Row Status textual convention, refer to Appendix E.

You can modify the entries in the Process Monitor table by issuing SNMP Set request messages from your NMS to the SystemEDGE agent. Each time the Process Monitor table is successfully modified, the agent updates the `/etc/sysedge.mon` file to record the changes. This enables the agent to start up with the same Process Monitor table configuration that it had when it was stopped. The `sysedge.mon` file preserves changes that are made during the operation of the agent across agent and system restarts. The agent *overwrites* the `/etc/sysedge.mon` or `%SystemRoot%\system32\sysedge.mon` configuration files every time the Process Monitor table is modified.

NOTE

Configuration file directives in `sysedge.cf` take precedence over entries in `sysedge.mon`. For example, if a Process Monitor table entry is contained in `sysedge.mon` at index 10, and a configuration file directive for index 10 of the Process Monitor table is added to `sysedge.cf`, the entry that is defined in `sysedge.cf` replaces the entry from `sysedge.mon`.

The SystemEDGE agent software distribution includes a command-line utility named `edgewatch` that takes a process name or PID as a command-line argument and dynamically configures an entry in the Process Monitor table to monitor the process. For instructions on how to use the `edgewatch` utility, refer to “Using the `edgewatch` Utility to Monitor Processes” on page 310.

Initial Configuration During Startup

On startup, the agent reads the `sysedge.cf` file. You can use this file to specify which MIB variables you want SystemEDGE to monitor. You can do so by adding watch process configuration file directives to the file. This directive automatically configures the agent to monitor an attribute of a process that you specify. You identify the process to be monitored through a regular expression that matches the process name and (optionally, for UNIX systems) its arguments. The agent automatically determines the PID for the specified process and then creates the appropriate entry in the agent’s Process Monitor table. You do not need to know the PID ahead of time or to use SNMP Set requests to add an entry to the Process Monitor table to use the watch process directive. In addition, if the process is not yet running, the agent continues to try to match the regular expression to a process until it succeeds. Then, it initiates process monitoring.

Selecting Processes and Attributes for Monitoring

This section describes how to specify MIB objects for process monitoring by adding them to the `sysedge.cf` file. As an alternative, you can specify MIB objects through SNMP Set operations. For more information about specifying objects through SNMP Set operations, refer to “Dynamic Configuration During Operation” on page 299.

To specify process attributes to monitor:

1. Select the process and attribute to monitor. The process and attribute you select must be supported by the SystemEDGE agent and implemented on the platform on which the agent is running. For a list of supported variables, refer to the MIB specifications (in the /doc subdirectory of the SystemEDGE agent distribution).
2. Assign the entry to a free row in the table by selecting the index number.
3. Decide on the sample type: if the attribute you want to monitor is a counter, use `deltaValue`. For other integer values (gauge, enumerated integer, integer, and so on), use `absoluteValue`.
4. Decide on the threshold and operator type against which the agent should compare the monitored variable's current value. The agent uses the following comparison expression the agent: *current-value operator value*
Valid values for *operator* are described in Table 37 on page 284.
5. Select an appropriate value for comparison. To help determine an appropriate value, monitor the object for a period of time to find a normal value. *The choice of this value is critical and depends on the semantics of the object you are monitoring.* If you want to receive `processClear` traps, enable that feature through the `pmonFlags` field. For more information, refer to Table 37 on page 284.
6. Select a monitor interval in seconds. The interval *must* be a multiple of 30 seconds. Choose the interval carefully. For example, do not set the agent to sample so frequently that an operator does not have time to act on the monitored condition if an exception occurs.

Alternatively, you could use a management station supporting the RowStatus operation to add the row via SNMP. Refer to “Dynamic Configuration During Operation” on page 299.

7. Write a configuration directive to define the process you have selected, and add it to the `sysedge.cf` file.
8. Start the SystemEDGE agent.

NOTE

If you are monitoring the `procAlive` process attribute, the SystemEDGE agent will automatically construct the appropriate Boolean expression.

Monitoring a Process to Make Sure It Is Running

You can use the `procAlive` process attribute to ensure that a process is up and running. When it is not running, the SystemEDGE agent sends a `processStop` trap and invokes an action. When the process is restarted, the agent sends a `processStart` trap.

Using the `watch process procAlive` Directive

You can use the `watch process procAlive` directive to add entries to the Process Monitor table as follows:

```
watch process procAlive 'procname' index flags interv 'description' 'action'
```

Table 42 describes the parameters to the watch process procAlive directive.

Table 42. Watch Process ProcAlive Parameters

Parameter	Description
<i>'procname'</i>	Quoted string that specifies the regular expression to apply when attempting to match a process name and optional arguments. NOTE: Because the Windows kernel does not track the arguments used in a process, the SystemEDGE agent does not match process arguments for Windows systems.
<i>index</i>	Row (index) to use for this entry.
<i>flags</i>	Specifies any additional, non-default behavior to apply to this entry. Specify all flags as hexadecimal numbers (for example, 0x0000).
<i>interv</i>	Interval that indicates how often (in seconds) the agent monitors the process.
<i>'description'</i>	Quoted string that is 0 to 128 characters in length and that normally contains a description of the process and attribute that the agent is monitoring, as well as severity level for this event.
<i>'action'</i>	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to execute when the process starts or stops. If the string is empty, the agent performs no action for this entry.

The watch process procAlive configuration file directive automatically creates the following boolean expression for the entry:

```
hrSWRunStatus = 4
```

If the status of the process (as determined by the Host Resources Running Software table) equals invalid(4), or if the process stops running, the agent sends a processStop SNMP trap. If an action is configured, the agent also invokes the action. If the process restarts, the agent automatically detects the new PID, reinitializes the corresponding entry, and sends a processStart SNMP trap.

When SystemEDGE is monitoring Windows services directly (as enabled through the Process Monitor table flag 0x08000), the watch process `procAlive` configuration file directive automatically creates the following boolean expression for the Process Monitor table entry:

```
ntServiceState ≠ 1
```

When the service's status (as determined by the NT Service table) becomes `notRunning` (its status does not equal 1), this expression evaluates to `True`, and the SystemEDGE agent sends a `processStop` SNMP trap. If an action is configured, the agent also invokes it. If the service restarts, the agent automatically detects that the service has restarted, and then reinitializes the corresponding Process Monitor Table entry and sends an `processStart` SNMP trap.

Using the watch process Directive to Monitor Process Attributes

You can use the watch process directive to configure the SystemEDGE agent to monitor any attribute of a process other than its liveness (`procAlive`). Use the watch process directive as follows:

```
watch process attribute 'procname' index flags interval stype operator  
threshold 'description' 'action'
```


Table 43 describes the parameters of the watch process directive.

Table 43. Watch Process Parameters (Page 1 of 2)

Parameter	Description
<i>attribute</i>	Process attribute that the SystemEDGE agent monitors for the specified threshold. You may select any process attribute (except <code>procAlive</code>) from Table 38 on page 288.
<i>'procname'</i>	Quoted string that specifies the regular expression to apply when attempting to match a process name and optional arguments. NOTE: Because the Windows kernel does not track the arguments used in a process, the SystemEDGE agent does not match process arguments for Windows systems.
<i>index</i>	Row (index) to use for this entry.
<i>flags</i>	Specifies any additional, non-default behavior to apply to this entry. Specify all flags as hexadecimal numbers (for example, <code>0x0000</code>).
<i>interval</i>	Interval that indicates how often (in seconds) the agent monitors the process. NOTE: This value must be a multiple of 30.
<i>stype</i>	Either absolute or delta; this value indicates whether the agent should sample the process attribute's absolute value, or take the difference between successive samples.
<i>operator</i>	Operator type. A Boolean operator that is used for evaluating the expression: <i>current-value operator value</i> The operator can be any of the following: <ul style="list-style-type: none"> • <code>nop</code> (no operation) • <code>></code> (greater than) • <code><</code> (less than) • <code>>=</code> (greater than or equal to) • <code><=</code> (less than or equal to) • <code>==</code> (equal) • <code>!=</code> (not equal)

Table 43. Watch Process Parameters (Page 2 of 2)

Parameter	Description
<i>threshold</i>	Integer value (threshold) to which the agent compares the current value (either absolute or delta).
' <i>description</i> '	Quoted string that is 0 to 128 characters in length and that normally contains a description of the process and attribute that are being monitored, as well as severity level.
' <i>action</i> '	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when the process starts or stops. If the string is empty, the agent performs no action for this entry.

The watch process configuration file directive automatically creates the following boolean expression for the Process Monitor table entry:

```
attribute operator threshold
```

When this expression evaluates to True, the SystemEDGE agent sends a processThreshold SNMP trap. If you have configured an action, the agent invokes it. If the process has died, the Process Monitor table entry becomes notReady, and the agent sends a notReady SNMP trap. However, if the process restarts, the agent automatically reinitializes the corresponding Process Monitor table entry, reacquires the PID, and continues monitoring that process attribute for the specified threshold.

Process Monitoring Examples

This section contains sample configuration file directives for process monitoring. Each example shows how to define an instance of process monitoring and explains the attribute or threshold that is being monitored.

Monitoring Sendmail to Make Sure It Is Running

The following example configures the SystemEDGE agent to monitor the sendmail daemon on the underlying system:

```
watch process procAlive 'sendmail' 11 0x00000100 60 'Monitor sendmail' ''
```

The values in this example are defined as follows:

- 11 indicates that this entry will occupy row 11 (pmonIndex=11) in the Process Monitor table.
- The flags field value of 0x00000100 indicates that the agent should monitor the parent sendmail process if more than one sendmail daemon is present and running.
- 60 indicates that the agent should check the sendmail process every 60 seconds.
- No action is specified, so the agent will not invoke a command when it sends a trap.

Monitoring the Simple TCP/IP Services Process To Make Sure It Is Running

The following example configures the SystemEDGE agent to monitor the TCPSVCS process that makes up the Simple TCP/IP Services service:

```
watch process procAlive 'TCPSVCS' 15 0x00000000 30 'Monitor NT TCP services' ''
```

The values in this example are defined as follows:

- 15 indicates that this entry will occupy row 15 (pmonIndex=15) in the Process Monitor table.
- The flags field value of 0x00000000 indicates that the agent should provide the default process-monitoring behavior.
- 30 indicates that the agent should check the *TCPSVCS* process every 30 seconds.
- No action is specified, so the agent will not invoke a command when it sends a trap.

NOTE

This example illustrates how to monitor the *underlying process* that provides the Windows Simple TCP/IP Services service. The following example illustrates how to monitor the Windows service itself rather than its underlying process.

Monitoring the Simple TCP/IP Services Service

Both of the following examples configure the SystemEDGE agent to monitor the TCPSVCS *service* itself rather than the underlying process:

```
watch process procAlive 'Simple TCP/IP Services' 15 0x08000 30 'Monitor NT  
TCP/IP Services' ''
```

-or-

```
watch ntservice 'Simple TCP/IP Services' 15 0x0 30 'Monitor NT TCP/IP  
Services' ''
```

The values in this example are defined as follows:

- 15 indicates that this entry will occupy row 15 (pmonIndex=15) in the Process Monitor table.
- The flags field value of 0x08000 indicates that the agent should monitor the Windows service, rather than the underlying process.
- 30 indicates that the agent should check the Simple TCP/IP Services service every 30 seconds.
- No action is specified, so the agent will not invoke a command when it sends a trap.

Monitoring ypbind To Make Sure It Is Running

The following example configures the SystemEDGE agent to monitor the UNIX ypbind daemon on the underlying system:

```
watch process procAlive 'ypbind' 16 0x00000000 60 'Monitor ypbind'  
'/example/pager.sh'
```

The values in this example are defined as follows:

- 16 indicates that this entry will occupy row 16 (pmonIndex=16) in the Process Monitor table.
- 60 indicates that the agent should check the ypbind process every 60 seconds.
- The agent invokes the specified action script /example/pager.sh each time it sends a trap. In this case, it invokes the script each time a processStop or a processStart trap is sent. The script should examine its arguments to determine which trap is being sent and then send the appropriate message to the target pager.

Monitoring the Size of a Process

The following example configures the SystemEDGE agent to monitor the overall size of a particular process:

```
watch process procSize 'netscape' 20 0x00a02400 60 absolute '>' 35000 'Monitor  
netscape size' ''
```

The values in this example are defined as follows:

- procSize is the attribute that the agent is monitoring. It returns the size of text, data, and stack segments of the corresponding process. Monitoring this attribute for a given threshold will allow you to determine if it is leaking memory or growing unbounded.
- netscape indicates the name of the process that the agent will monitor.
- 20 indicates that this entry will occupy row 20 (pmonIndex=20) of the Process Monitor table.

- The flags field of 0x00a02400 instructs the agent to modify the default Process Monitor table behavior as follows:
 - 0x00000400 instructs the agent to send processClear traps.
 - 0x00002000 instructs the agent to send up to 10 consecutive traps and then send no more.
 - 0x00a00000 contains the flag value 10 for use with the directive in this example.
- The sample type absolute instructs the agent to compare each sampled value to the threshold rather than to measure the difference (delta) between successive samples.
- The greater than (>) operator instructs the agent to compare the sampled procSize attribute against the value 35000 (35,000 KB or 35 MB), and to send a processThreshold trap when that threshold is exceeded.

Using the edgewatch Utility to Monitor Processes

edgewatch is a command-line utility that automatically configures the SystemEDGE agent to monitor processes, log files, and Windows event logs. After you specify the particular process, log file, or Windows event log, as well as the associated arguments, the edgewatch utility issues an SNMP Set request to create the appropriate entry in the target agent's self-monitoring table.

You can use the edgewatch utility for process monitoring as follows:

```
edgewatch hostname [:port][,timeout] community process command
```

Table 44 describes the edgewatch parameters.

Table 44. edgewatch Parameters

Parameter	Description
<i>hostname[:port][,timeout]</i>	Specifies the hostname or IP address (in dotted notation) of the system on which the agent exists. If the agent is running on an alternative UDP port (for example, 1691), specify that port number through a colon-separator. In addition, you can specify an optional SNMP timeout value (in seconds) using a command separator.
<i>community</i>	Specifies the community string that edgewatch uses in its SNMP requests to the agent. Because edgewatch uses SNMP Set requests, the community string should provide read-write access to the target agent.
<i>command</i>	<p>Specifies the command and associated arguments. Supported commands include the following:</p> <ul style="list-style-type: none">• add• setstatus• delete• list <p>Those commands are outlined in the next section.</p>

edgwatch Commands for Process Monitoring

The edgwatch process-monitoring commands and associated arguments are as follows:

```
add procAlive processname index flags interval 'description' 'action'
add attribute processname index flags interval stype operator value
'description' 'action'
setstatus index status
delete index
list
```

Table 45 defines the arguments to the edgwatch commands.

Table 45. edwatch Command Arguments (Page 1 of 2)

Argument	Description
processname	Specifies the regular expression used to find the PID of the process to monitor. Enclose this value in quotation marks if it contains spaces or other special characters.
index	Specifies the row (index) to use for this monitoring entry.
flags	Specifies the hexadecimal flags (for example, 0x00000001) that indicate the additional behavioral semantics of this entry.
interval	Specifies an integer value (30 to MAXINT) that indicates how often (in seconds) the agent should monitor the process. For example, the value 30 instructs the agent to monitor this entry every 30 seconds. NOTE: This value must be a multiple of 30 seconds.
'description'	Specifies a quoted string that is 0 to 128 characters in length and that normally contains a description of the process and attribute that are being monitored, as well as a severity level for this event.
'action'	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when the expression evaluates to True and a trap is sent. If the string is empty, the agent invokes no action for this entry.

Table 45. edwatch Command Arguments (Page 2 of 2)

Argument	Description
status	One of the following: <ul style="list-style-type: none"> • active (activate a row) • notInService (deactivate but preserve a row) • destroy (delete a row)
attribute	Process attribute that the agent monitors for the given threshold. You may choose any process attribute from Table 38 on page 288. The arguments for procAlive differ from those for the other attributes. For more information, refer to “Using the watch process procAlive Directive” on page 302.

edgwatch Examples

This section includes examples for using edgwatch.

Monitoring the ypbind Process

The following example creates a Process Monitor table entry at index 16 to monitor the ypbind process that is running on the target system:

```
edgwatch 143.45.0.12 private process add procAlive "ypbind" 16 0x00 60
"Monitor ypbind" "/example/pager.sh"
```

The values in this example are defined as follows:

- procAlive is the process attribute being monitored. It instructs the agent to monitor the process to make sure it is running. ypbind is the process that the agent is monitoring. It is responsible for client directory lookups and is necessary for machines running Network Information Services (NIS).
- The flags field 0x00 instructs the agent to provide the default behavior for this table entry.
- If the process dies, the agent sends a processStop trap and runs the action script /example/pager.sh.

Monitoring the netscape Process

The following example creates a Process Monitor table entry at index 20 to monitor the netscape process that is running on the target machine:

```
edgewatch 143.45.0.12 private process add procSize "netscape" 20 0x00a02400 60  
absolute ">" 35000 "Monitor netscape size" ""
```

The values in this example are defined as follows:

- `procSize` is the process attribute that is being monitored. It instructs the agent to monitor the overall size of the program's text, data, and stack segments.
- `netscape` is the application that the agent is monitoring.
- The flags field of `0x00a02400` instructs the agent to modify the default Process Monitor table behavior as follows:
 - `0x00000400` instructs the agent to send `processClear` traps.
 - `0x00002000` instructs the agent to send up to 10 consecutive traps and then send no more.
 - `0x00a00000` contains the flag value 10 for use with this directive.
- The greater than (`>`) operator indicates that an event should occur when the process size of netscape exceeds the threshold (35 MB).
- The threshold is 35,000 KB or 35 MB.

Monitoring the Windows TCPSVCS Process

The following example creates a Process Monitor table entry at index 15 to monitor the Windows *TCPSVCS* process (or service) that is running on the target system:

```
edgewatch 143.45.0.12 private process add procAlive "TCPSVCS" 15 0x00 30  
"Monitor NT TCP services" ""
```

The values in this example are defined as follows:

- `procAlive` is the process attribute that is being monitored. It instructs the agent to scan the Process Monitor table periodically (every 30 seconds) to ensure that this process is running.
- `TCPVCS` is the Windows service that is responsible for TCP-related services on Windows systems.
- The `flags` field of `0x00` instructs the agent to provide the default behavior for this table entry.

Removing Process Monitoring Entries

To stop the self-monitoring of a particular process attribute, you must remove the appropriate entry from the Process Monitor table. The `watch process` directives in the `sysedge.cf` file creates a Process Monitor table entry whenever the SystemEDGE agent starts. This row creation results in a new Process Monitor table entry that is stored in `sysedge.mon`. Thus, permanent removal of a Process Monitor table entry requires two steps:

1. Remove the entry from the `sysedge.cf` file.
2. Remove the entry from the Process Monitor table.

Removing Entries from the `sysedge.cf` File

If you configured a Process Monitor table entry by adding a `watch process` directive to the `sysedge.cf` file, you must remove it from that file as part of removing the entry from the table. If you do not remove the `sysedge.cf` directive, the entry will be recreated the next time the SystemEDGE agent starts.

Removing Entries with the edgemon Utility

To remove a process-monitoring entry from the Process Monitor table, use the edgemon utility to delete the entry. The following example deletes row 14 from the Process Monitor table on host 143.45.0.12. Once deleted, the row will be removed both from memory and from the sysedge.mon file:

```
edgemon 143.45.0.12 private process delete 14
```

Removing Entries Manually

In some cases it may not be possible to use the edgemon utility to delete Process Monitor table entries. For example, if you have configured the SystemEDGE agent to disallow SNMP Set operations, the edgemon utility does not work. In this case, you must remove the entry from the Process Monitor table by editing the sysedge.mon file to remove the entry. Because this is an active file, you must stop the SystemEDGE agent before editing the file. For more information on the format of this file, refer to Appendix C.

For example, to delete row 14:

1. Stop the SystemEDGE agent.
2. Open sysedge.mon for editing, delete the entry for processmon row 14, and save the file.
3. Open sysedge.cf for editing, delete the entry for processmon row 14 if it exists, and save the file.
4. Restart the SystemEDGE agent.

Recommendations for Process and Service Monitoring

When you are configuring process and service monitoring, consider the following:

- Monitor the following:
 - Status of the processes:
 - Use the `processState` (operating-system dependent) or `processStateStr` (operating-system independent) OIDs to return the process state. For more information, refer to `empire.asn1`.
 - For processes that run through an interpreter such as Perl, set the `0x00000800` flag to match the process name and arguments (UNIX only) when creating process-monitoring entries. For more information, refer to Table 40 on page 291.
 - CPU time over interval (if this value is not incrementing, the process might be in the zombie state.)
 - Total CPU time (a value that is too high can indicate problems)
 - Leaking memory (RSS over time; use alarm thresholds to notify you of problems)
- Configure `SystemEDGE` to automatically restart failed processes.
- Set your UNIX application shutdown files to disable process and service monitoring to prevent race conditions.
- Use the Process Group Monitor table to monitor multiple processes with the same name. For more information, refer to Chapter 12, “Configuring Process and Service Monitoring.”

Configuring Process Group Monitoring

This chapter explains how to use the SystemEDGE agent to monitor groups of processes. The agent uses process groups to aggregate the per-process information into a single, easy-to-poll, easy-to-monitor value.

Monitoring Process Groups

For information about monitoring individual processes and Windows services, refer to Chapter 11, “Configuring Process and Service Monitoring.”

The flexible Process Group Monitor table of the Systems Management MIB enables you to configure the SystemEDGE agent dynamically to monitor groups of processes that are running on the underlying system. You select the process group, regular expression, and interval, and the agent uses that information to monitor those process groups. You can configure the SystemEDGE agent to monitor process groups to determine what processes exist in each group and whether the group membership changes. If components of an application start or fail, or if members leave a group or are added to a group, the SystemEDGE agent can automatically notify the NMS.

The Process Group Monitor Table

The Process Group Monitor table provides information about a group of processes that the agent is currently monitoring. You can add entries to this table, modify existing entries, or remove entries from the table. The SystemEDGE agent sends a processGroupChange trap to indicate when a process group changes, and you can configure the agent to perform actions whenever a group changes.

Columns of the Process Group Monitor Table

Table 46 describes the columns that make up the Process Group Monitor table. For a complete description of the Process Group Monitor table and its fields, refer to the Systems Management MIB specification (empire.asn1 in the /doc subdirectory of the SystemEDGE agent distribution).

NOTE

SystemEDGE maintains a history of group membership and tracks process statistics even after an individual process has left the group.

Table 46. Columns of the Process Group Monitor Table (Page 1 of 4)

Column Name	Permissions	Description
pgmonIndex	Read-Only	Integer (1 to MAXINT) that indicates the row index for this entry. Index values must be unique, but they do not need to be contiguous.
pgmonDescr	Read-Write	Quoted string that is 0 to 128 characters in length and that normally contains a description of the entry and who created it.
pgmonInterval	Read-Write	Integer value (1 to MAXINT) that indicates how often (in seconds) the agent should sample the variable. For example, the value 30 instructs the agent to monitor this entry every 30 seconds. NOTE: This value must be a multiple of 30 seconds.

Table 46. Columns of the Process Group Monitor Table (Page 2 of 4)

Column Name	Permissions	Description
pgmonProcRegExpr	Read-Write	Regular expression to apply when the agent is attempting to match processes by name.
pgmonFlags	Read-Write	Integer flags value that dictates the behavior of each entry. The default is 0x00. For more information about this field, refer to “Process Group Monitor Table Flags” on page 325.
pgmonNumProcs	Read-Only	Current number of processes in the process group that this entry is tracking. A process belongs to the process group if its name (and possibly, its arguments) match the regular expression that was configured for this entry.
pgmonPIDList	Read-Only	List of the numeric PIDs in this process group. Each PID is separated from the next by a space character.
pgmonStatusList	Read-Only	List of the process states in this process group. Each process state is separated from the next with a space character. Entries in the status list have a one-to-one correspondence with entries in the PID list. For more information about states of a process, refer to the processStateStr MIB variable.
pgmonAction	Read-Write	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when the expression evaluates to True. If the string is empty, the agent invokes no action for this entry. By default, it invokes no action.
pgmonNumEvents	Read-Only	Number of events for this entry. Events do not necessarily imply traps; traps can be turned off for the row through a flags setting.
pgmonNumTraps	Read-Only	Number of traps that the agent has sent for this entry.
pgmonLastTrap	Read-Only	Time (based on sysUpTime) at which the agent last sent a trap for this entry. A value of 0 indicates that no traps have been sent.

Table 46. Columns of the Process Group Monitor Table (Page 3 of 4)

Column Name	Permissions	Description
pgmonRowStatus	Read-Write	<p>One of the following:</p> <ul style="list-style-type: none"> • active(1) • notInService(2) • notReady(3) • createAndGo(4) • createAndWait(5) • destroy(6) <p>Normally, a row is either active or notInService. These values are identical in meaning to the SNMPv2 SMI RowStatus textual convention. For more information, refer to Appendix E.</p>
pgmonRSS	Read-Only	<p>Combined resident set size (RSS) of the group of processes. The RSS for each process in the group is summed at each interval and stored in this variable. Because RSS also includes shared memory, the total RSS for a group could exceed total possible physical memory for the underlying system. For more information, refer to the processRSS variable of the Systems Management MIB (in /doc/empire.asn1).</p>
pgmonSize	Read-Only	<p>Combined size of the text, data, and stack segments of the group of processes. The size of each process in the group is summed at each interval and stored in this variable. Size includes shared memory, so the total size for a group could exceed the total virtual memory for the underlying system. For more information, refer to the processSize variable of the Systems Management MIB (in /doc/empire.asn1).</p>
pgmonThreadCount	Read-Only	<p>Total number of threads for the group of processes. The number of threads running in each process in the group is summed at each interval and stored in this MIB variable. For more information, refer to processNumThreads variable of the Systems Management MIB (in /doc/empire.asn1).</p>

Table 46. Columns of the Process Group Monitor Table (Page 4 of 4)

Column Name	Permissions	Description
pgmonMEM	Read-Only	Total percentage of real memory that is being used by the processes in this group. The percentage of memory being used is summed at each interval and stored in this MIB variable. Memory usage includes shared memory (shared libraries and DLLs), so the total percentage may exceed 100.
pgmonInBlks	Read-Only	Number of blocks of data input by processes in this group.
pgmonOutBlks	Read-Only	Number of blocks of data output by processes in this group.
pgmonMsgsSent	Read-Only	Number of messages sent by processes in this group.
pgmonMsgsRecv	Read-Only	Number of messages received by processes in this group.
pgmonSysCalls	Read-Only	Number of system calls invoked by processes in this group.
pgmonMinorPgFlts	Read-Only	Number of minor page faults incurred by processes in this group.
pgmonMajorPgFlts	Read-Only	Number of major page faults incurred by processes in this group.
pgmonNumSwaps	Read-Only	Number of times processes in this group have been swapped.
pgmonVolCtx	Read-Only	Number of voluntary context switches incurred by processes in this group.
pgmonInvolCtx	Read-Only	Number of involuntary context switches incurred by processes in this group.
pgmonCPUSecs	Read-Only	Number of seconds of CPU time used by processes in this group.
pgmonMatchUser	Read-Write	When this object is set to a valid user name, SystemEDGE matches running processes by user name, as well as any process name regular expression. This variable is valid only on UNIX systems.
pgmonMatchGroup	Read-Write	When this object is set to a valid group name, SystemEDGE matches running processes by group name, as well as process name regular expression and user name. This variable is valid only on UNIX systems.

Figure 46 shows a sample Process Group Monitor table.

The screenshot shows a window titled 'Table: processGroupMonTable Path: 1.3.6.1.4.1.1546.15.10.1 Rows:3 Columns:27'. The table contains the following data:

Instance	pgmonIndex	pgmonDescr	pgmonInterval	pgmonProcRegExpr	pgmonFlags	pgmonNumProcs
1	1	Watch NFS Server	60	nfsd	0	8
2	2	Watch Web Server	60	httpd	0	11
3	3	Watch SSH Server	60	sshd	0	2

At the bottom of the window are 'OK' and 'Next-Page' buttons.

Figure 46. Sample Process Group Monitor Table

Optimizing Row Creation

You can use the following MIB objects with the Process Group Monitor table to optimize row creation.

Table 47. Scalar Objects for Optimizing Row Creation

MIB Object	Permissions	Description
pgmonUnusedIndex	Read-Only	You can use this variable to optimize table-row creation for the Process Group Monitor table. Perform an SNMP Get of this variable to return an unused index number for the Process Group Monitor table.
pgmonMatchDescr	Read-Write	You can use this MIB object with the pgmonMatchIndex MIB object to determine the index number that corresponds to a particular entry description. Perform an SNMP Set of this MIB object to cause the agent to search through entries in the Process Group Monitor table and place the index value of the last entry whose description matches in the pgmonMatchIndex MIB object.
pgmonMatchIndex	Read-Only	You can use this MIB object with pgmonMatchDescr to match a particular entry description with its index number.

Process Group Monitor Table Flags

The **pgmonFlags** column in the Process Group Monitor table is a 32-bit unsigned integer field that can specify additional behavioral semantics for the corresponding row.

By default, the Process Group Monitor table row does the following:

- Attempts to reinitialize itself
- Sends SNMP traps
- Logs syslog events
- Invokes actions (if they are configured)

You can set different flag bits to alter these defaults. The agent interprets all flags in hexadecimal (base 16) notation. Figure 47 shows the composition of the Process Group Monitor Table flags (pgmonFlags) field.

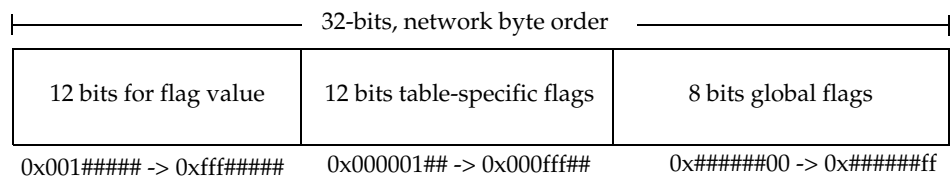


Figure 47. Process Group Monitor Table Flags

The flags value consists of three fields:

- **Field 1:** Common table flags that are defined for the self-monitoring tables of the Systems Management MIB. This portion is the low-order 8 bits of the flags field.
- **Field 2:** Table-specific flags that are defined separately for each of the self-monitoring tables. This field defines the next 12 low-order bits after the common table flags.
- **Field 3:** Reserved 12 high-order bits for an integer value for use in conjunction with table-specific flags. Flags in the Process Group Monitor table-specific flags field define the contents of this field.

The following sections explain each flag bit. You can combine flag values through a logical OR operation. One flag is specific to the Process Group Monitor table: 0x00100. This flag instructs the SystemEDGE agent to match the process name and arguments for this entry. Table 48 describes the Process Monitor table flags.

Table 48. Process Group Monitor Table Flags

Flag	Description
0x00000001	Do not execute actions for this entry.
0x00000002	Do not send traps for this entry. This flag bit overrides any other flag bit with respect to traps.
0x00000004	Do not attempt to reinitialize this entry. By default, the agent periodically tries to reinitialize this table entry by scanning the process table to determine the new process ID if the target process has been restarted. Setting this bit disables automatic reinitialization.
0x00000008	Do not record events for this entry in the syslog facility. Setting this bit does not affect trap sending or threshold monitoring. It prevents the event from being logged through syslog only. On Windows systems, it prevents logging events to the agent's log file (sysedge.log). When events occur frequently, it may be useful to disable event logging.
0x00000020	Do not pass SystemEDGE arguments to action scripts or programs. SystemEDGE normally passes default action parameters that indicate the trap type, description field, and so on. This flag disables the passing of those arguments. For more information, refer to “Process Group Monitor Table Actions” on page 326.
0x00000040	Do not send notReady traps for this entry.
0x00000100	Match the process name and arguments for this entry.

Process Group Monitor Table Actions

The SystemEDGE agent provides several default parameters to the action commands when they are invoked. These parameters are in addition to any parameters that you specify in the action string and are passed on the command line *after* those that you

specify. The default parameters are the same as the parameters provided in the SNMP traps that are sent for the Monitor table. For more information about traps sent by the SystemEDGE agent, refer to Chapter 8, “Private Enterprise Traps.”

Viewing the Process Group Monitor Table with AdvantEDGE View

If you are using AdvantEDGE View, you can query a system for Process Group Monitor table information by selecting the system you want to monitor from the **System** list, selecting **Process Group Monitoring** from the **Configuration** list, and clicking the **Configuration** icon. For more information, refer to the AdvantEDGE View Web Help. Figure 48 shows a sample AdvantEDGE View Process Group Monitor table.

Index	Description	Interval	RegExpr	User	Group	Flags	Action	Row Status	Num Procs
9	Monitor NFS Server	60	nfs	admin	(any)	0x10b	(no action)		2
10	Monitor Web Server	60	http	(any)	(any)	0xa	(no action)		0
11	Monitor SSH Server	30	sshd	(any)	(any)	0x0	(no action)		0
12	Monitor FTP Server	30	ftpd	admin	(any)	0x0	(no action)		0
13	Monitor all	60	.*	(any)	(any)	0xb	(no action)		35
14	Monitor RPC daemons	60	rpc	(any)	(any)	0xa	(no action)		1

Figure 48. Sample AdvantEDGE View Process Group Monitor Table

Assigning Entry Rows for the Process Group Monitor Table

The **pgmonIndex** column of the Process Group Monitor table acts as a key field (or row index) to distinguish rows in the table. Rows 1 through 10 are reserved for internal use by the SystemEDGE agent. Users can configure rows in the range of 11 to MAXINT. For more information about reserving blocks of rows, refer to “Reserving Blocks of Rows” on page 258.

Configuring the Process Group Monitor Table

You can control which processes and process attributes the SystemEDGE agent monitors by adding, deleting, or modifying the entries in the Process Group Monitor table. You can configure the Process Group Monitor table in one of these ways:

- **Dynamically.** Use SNMP commands from a management system, such as AdvantEDGE View, to modify the table. For more information, refer to the next section, “Dynamic Configuration During Operation.”
- **At start-up initialization.** Specify the process attributes to monitor through the agent’s configuration file `sysedge.cf`. For more information, refer to “Initial Configuration During Startup” on page 329.

Dynamic Configuration During Operation

The SystemEDGE agent uses the SNMPv2 SMI Row Status textual convention for creating, deleting, and modifying rows in the table. For more information about the Row Status textual convention, refer to Appendix E.

You can use your NMS platform to issue SNMP Set request messages to the SystemEDGE agent to modify the entries in the Process Group Monitor table. Each time a Set request successfully modifies the table, the agent updates the `sysedge.mon` file to record the changes so that the agent starts up with the same Process Group Monitor table configuration it had when it was stopped. That is, the agent *overwrites* the `/etc/sysedge.mon` or `%SystemRoot%\system32\sysedge.mon` configuration files every time the Process Group Monitor table is modified. Any changes made during the operation of the agent are preserved in this file across agent and system restarts.

NOTE

Configuration file directives in `sysedge.cf` take precedence over entries in `sysedge.mon`. If, for example, a Process Group Monitor table entry is contained in `sysedge.mon` at index 10, and a configuration file directive is added to `sysedge.cf` at index 10 for the Process Group Monitor table, the entry defined in `sysedge.cf` replaces the entry in `sysedge.mon`.

Initial Configuration During Startup

On startup, the agent reads the `sysedge.cf` file. You can use this file to specify which MIB variables you would like SystemEDGE to monitor. You can do so through the **watch procgroup** configuration file directive. You can identify the process group to be monitored through a regular expression that matches the process name and (optionally) its arguments.

Monitoring a Process Group

You can use the `watch procgroup` directive to monitor a process group as follows:

```
watch procgroup 'regexp' index flags interval 'description' 'action'
```

Table 49 describes the parameters for `watch procgroup`.

Table 49. watch procgroup Parameters

Parameter	Description
<code>'regexp'</code>	Quoted string that specifies the regular expression to apply when attempting to match a process name and optional arguments.
<code>index</code>	Row (index) to use for this entry.
<code>flags</code>	Specifies any additional, non-default behavior to apply to this entry. Specify all flags as hexadecimal numbers (for example, 0x0000).
<code>interval</code>	Interval that indicates how often (in seconds) the agent monitors the process group.
<code>'description'</code>	Quoted string that is 0 to 128 characters in length and that normally contains a description of the process group that is being monitored.
<code>'action'</code>	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to execute when a match is found for the process group. If the string is empty, the agent invokes no action for this entry.

Process Group Monitoring Examples

This section contains sample configuration file directives for process group monitoring.

Monitoring the xterm Process Group

The following example configures the SystemEDGE agent to monitor the xterm process group:

```
watch procgroup 'xterm.*' 10 0x00 60 'Watch xterms' ''
```

The values in this example are defined as follows:

- 10 indicates that this entry will occupy row 10 (pgmonIndex=10) in the Process Group Monitor table.
- The flags field value of 0x00 indicates the default behavior.
- 60 indicates that the agent should check the xterm process group every 60 seconds.
- No action is specified, so the agent invokes no command when it sends a trap.

Monitoring the emacs Process Group

The following example configures the SystemEDGE agent to monitor the emacs process group:

```
watch procgroup 'emacs.*|xmibmgr' 11 0x00 60 'Watch emacs' ''
```

The values in this example are defined as follows:

- 11 indicates that this entry will occupy row 11 (pgmonIndex=11) in the Process Group Monitor table.
- The flags field value of 0x00 indicates the default behavior.
- 60 indicates that the agent should check the emacs process group every 60 seconds.
- No action is specified, so the agent invokes no command when it sends a trap.

Monitoring the DT Process Group

The following example configures the SystemEDGE agent to monitor the DT process group:

```
watch procgroup 'dt/bin' 12 0x00 60 'Watch DT stuff' ''
```

The values in this example are defined as follows:

- 12 indicates that this entry will occupy row 12 (pgmonIndex=12) in the Process Group Monitor Table.
- The flags field value of 0x00 indicates the default behavior.
- 60 indicates that the agent should check the DT process group every 60 seconds.
- No action is specified, so the agent invokes no command when it sends a trap.

Removing Process Group Monitoring Entries

To stop the self-monitoring of a particular process group, you must remove the appropriate entry from the Process Group Monitor table. The watch procgroup directives in the sysedge.cf file will create a Process Group Monitor table entry whenever the SystemEDGE agent is started. This row creation results in a new Process Group Monitor table entry that will be stored in sysedge.mon. Thus, permanent removal of a Process Group Monitor table entry requires two steps:

1. Remove the entry from the sysedge.cf file.
2. Remove the entry from the Process Group Monitor table.

Removing Entries from the sysedge.cf File

If you configured a Process Group Monitor table entry by adding a watch procgroup directive to the sysedge.cf file, you must remove it from that file as part of removing the entry from the table. If you do not remove the sysedge.cf directive, the entry will be recreated the next time the SystemEDGE agent is restarted.

Removing Entries with the edgemon Utility

To remove a process group monitoring entry from the Process Group Monitor table, use the edgemon utility to delete the entry. The following example deletes row 14 from the Process Group Monitor table on host 143.45.0.12. Once deleted, the row will be removed both from memory and from the `sysedge.mon` file:

```
edgemon 143.45.0.12 private procgroup delete 14
```

Removing Entries Manually

In some cases it may not be possible to use the edgemon utility to delete Process Group Monitor table entries. For example, if you have configured the SystemEDGE agent to disallow SNMP Set operations, the edgemon utility will not work. In this case, you must remove the entry from the Process Group Monitor table by editing the `sysedge.mon` file to remove the entry. Because this is an active file, you will need to stop the SystemEDGE agent before editing the file. For more information on the format of this file, refer to Appendix C.

For example, to delete row 14:

1. Stop the SystemEDGE agent.
2. Open `sysedge.mon` for editing, delete the entry for `procgroupmon` row 14, and save the file.
3. Open `sysedge.cf` for editing, delete the entry for `procgroupmon` row 14 if it exists, and save the file.
4. Restart the SystemEDGE agent.

Configuring Log File Monitoring

This chapter explains how to use the SystemEDGE agent to monitor log files for regular expressions.

Monitoring Log Files

The SystemEDGE agent can monitor ASCII-based text files continuously for the appearance of user-specified regular expressions. Whenever a match for the regular-expression is written to the log file the agent is monitoring, the agent notifies the management system with a trap message.

The flexible Log Monitor table of the Systems Management MIB enables you to configure the agent dynamically to monitor a log file for the regular expressions that you specify. Each entry in the table represents the monitoring of a single log file for a particular regular expression.

This log file monitoring provides a very flexible solution for monitoring applications by monitoring the messages that the applications log. This feature is also useful for security management; for example, you can configure the agent to monitor system log files for su messages to notify you of possible security violations.

When the agent starts (or after rows have been added to the Log Monitor table), it checks the status of (stats) each log file for its current length and last access time. Thereafter, the agent periodically stats each log file for additions or modifications

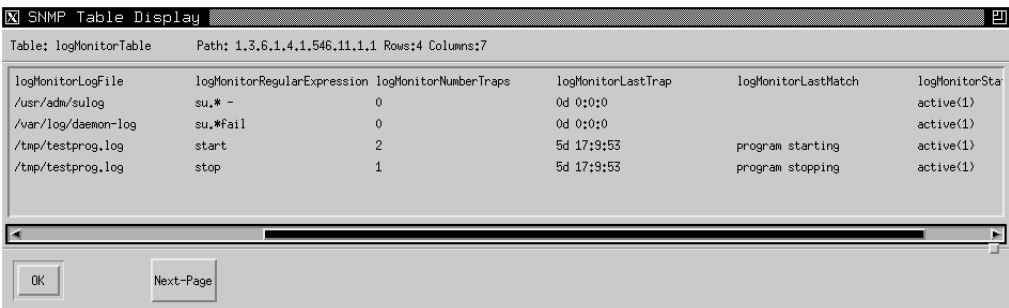
since the last status check. If the log file has changed, the agent scans *only* the changes—not the entire log file—to see if there is a match for the specified regular expression. If the agent finds a match, it sends the enterprise-specific logMonMatch SNMP trap to the configured NMS and executes the specified action for the row if the action field is not null. For more information about the logMonMatch trap, refer to “logMonMatch Trap” on page 222.

Log Monitor Table

For each entry, the Log Monitor table provides the following information:

- Name of the log file that the agent is monitoring for a particular regular expression
- Number of times a trap has been sent because a match was found
- Time at which the last trap was sent
- Log entry that caused the last match

Figure 49 shows sample entries for the Log Monitor table.



The image shows a window titled "SNMP Table Display". Inside, it displays the "logMonitorTable" with the path "1.3.6.1.4.1.546.11.1.1". The table has 7 columns: logMonitorLogFile, logMonitorRegularExpression, logMonitorNumberTraps, logMonitorLastTrap, logMonitorLastMatch, and logMonitorStatus. There are 4 rows of data.

logMonitorLogFile	logMonitorRegularExpression	logMonitorNumberTraps	logMonitorLastTrap	logMonitorLastMatch	logMonitorStatus
/usr/adm/sulog	su.*	0	0d 0:0:0		active(1)
/var/log/daemon-log	su.*Fail	0	0d 0:0:0		active(1)
/tmp/testprog.log	start	2	5d 17:9:53	program starting	active(1)
/tmp/testprog.log	stop	1	5d 17:9:53	program stopping	active(1)

At the bottom of the window are "OK" and "Next-Page" buttons.

Figure 49. Systems Management MIB: Log Monitor Table Sample Entries

Columns of the Log Monitor Table

Table 50 describes the columns of the Log Monitor Table. For more information about the Log Monitor Table and its fields, refer to the Systems Management MIB (empire.asn1 in the /doc subdirectory of the SystemEDGE agent distribution).

Table 50. Columns of the Log Monitor Table (Page 1 of 2)

Column Name	Permissions	Description
LogMonitorIndex	Read-Only	Row of the table.
LogMonitorLogFile	Read-Write	Complete path and file name of the log file to be monitored. NOTE: The file you monitor must be an ASCII-based text file. SystemEDGE does not support monitoring of other character sets, such as Unicode. You can determine a file's encoding by opening it in a text editor and selecting Save As . The encoding is listed in the Save as type field.
LogMonitorRegularExpression	Read-Write	Regular expression to search for when scanning the log file for matches. For information about the rules for specifying regular expressions, refer to the UNIX man page on egrep(1).
LogMonitorNumberTraps	Read-Only	Number of times that a trap was sent because a string matching the regular expression was logged to the file.
LogMonitorLastTrap	Read-Only	Time, based on sysUpTime, at which the agent last sent a trap for this entry.
LogMonitorLastMatch	Read-Only	Last log file entry that matched the regular expression; this variable is updated each time a match occurs.

Table 50. Columns of the Log Monitor Table (Page 2 of 2)

Column Name	Permissions	Description
LogMonitorStatus	Read-Write	SNMPv2 RowStatus; one of the following: <ul style="list-style-type: none">• active(1)• notInService(2)• notReady(3)
LogMonitorLogFileSize	Read-Only	Current size in bytes of the file that is being monitored.
LogMonitorLogFileLastUpdate	Read-Only	Time that the file was last updated.
LogMonitorDescr	Read-Write	Quoted string that is 0 to 128 characters in length and normally contains a description of the file being monitored, as well as a severity level for this event.
LogMonitorAction	Read-Write	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command, along with any parameters, to be executed when the regular expression is matched and a trap is sent. If the string is empty, the agent invokes no action for this entry.
LogMonitorFlags	Read-Write	Unsigned integer flags value indicating additional behavior that this row should follow during the course of its operation. By default, this field is assigned the hexadecimal value 0x00. For more information about this field, refer to the next section, “Log Monitor Table Flags.”

Optimizing Row Creation

You can use the following MIB objects with the Log Monitor table to optimize row creation.

Table 51. Scalar Objects for Optimizing Row Creation

	Permissions	Description
logmonUnusedIndex	Read-Only	You can use this variable to optimize table-row creation for the Log Monitor table. Perform an SNMP Get of this variable to return an unused index number for the Log Monitor table.
logmonMatchDescr	Read-Write	You can use this MIB object with the LogMonitorMatchIndex MIB object to determine the index number that corresponds to a particular entry description. Perform an SNMP Set of this MIB object to cause the agent to search through entries in the Log Monitor table and place the index value of the last entry whose description matches in the LogMonitorMatchIndex MIB object.
logmonMatchIndex	Read-Only	You can use this MIB object with LogMonitorMatchDescr to match a particular entry description with its index number.

Log Monitor Table Flags

The logMonitorFlags column in the Log Monitor table is a 32-bit unsigned integer that can specify additional behavior for the corresponding Log Monitor table row. By default, the Log Monitor table row does the following:

- Attempts to reinitialize itself
- Sends SNMP traps
- Logs matches to syslog
- Invokes actions (if they are configured)

You can specify different flag bits to alter these defaults. The SystemEDGE agent interprets all flags in hexadecimal (base 16) notation. Figure 50 shows the flags field (logMonitorFlags).

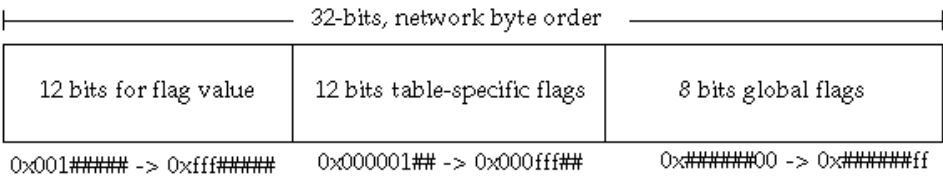


Figure 50. Log Monitor Table Flags

The flags consist of three fields:

- **Field 1:** Common table flags defined for the self-monitoring tables of the Systems Management MIB. This portion is the low-order 8 bits of the flags.
- **Field 2:** Table-specific flags that are defined separately for each of the self-monitoring tables. This field defines the next 12 low-order bits after the common table flags. Figure 51 on page 340 describes how these 12 bits are defined for the Log Monitor table.
- **Field 3:** Reserved 12 high-order bits for an integer value for use in conjunction with table-specific flags. This field includes flags that are specific to the Log Monitor table.

NOTE

The following sections define each flag bit. You can combine flag values through a logical OR operation.

Table 52 describes the flags of the Log Monitor table.

Table 52. Log Monitor Table Flags (Page 1 of 2)

Flag	Description
0x00000001	Do not execute actions for this entry.
0x00000002	Do not send traps for this entry. This flag bit overrides any other flag bit with respect to traps.
0x00000004	Do not attempt to reinitialize this entry. By default, if the monitored log file is ever unavailable, the agent will periodically try to reinitialize this table entry. Setting this bit disables automatic reinitialization.
0x00000008	Do not record events for this entry in the syslog facility. Setting this bit will not affect trap sending nor action execution. On Windows systems, the agent does not log events to the agent's log file (sysedge.log) if this bit is set. When events occur frequently, it is useful to disable event logging.
0x00000010	Send continuous logMonEntryNotReady traps for this entry every time the agent attempts to reinitialize logfile monitoring and fails. The agent's default behavior is to send a single logMonEntryNotReady trap when the log file that is being monitored ceases to exist, or when an error accessing that log file occurs. The agent periodically attempts to reinitialize the entry. Enabling this feature causes the agent to send an additional logMonEntryNotReady trap each time reinitialization fails.
0x00000020	Do not pass SystemEDGE arguments to action scripts or programs. SystemEDGE normally passes default action parameters that indicate the trap type, description field, and so on. This flag disables the passing of those arguments. For more information, refer to “Log Monitor Table Actions” on page 340.
0x00000040	Do not send notReady traps for this entry.

Table 52. Log Monitor Table Flags (Page 2 of 2)

Flag	Description
0x00000100	Apply the logical NOT operator to the regular-expression evaluation. If the regular expression evaluation equals False, this flag bit will be set to True and cause an event to occur. If the regular expression evaluation equals True, this flag bit will be set to False. Use caution when utilizing this capability because False evaluations are converted to True, and vice versa.
0x00000200	Track the log file's size, but do not parse through the file.
0x###00000	Flag value 0x00100000 → 0xffff0000. The Log Monitor table does not utilize this flag value field; setting it has no effect on the Log Monitor table operation or on any of the supported flag bits.

Figure 51 shows flag bits that are specific to the Log Monitor Table.

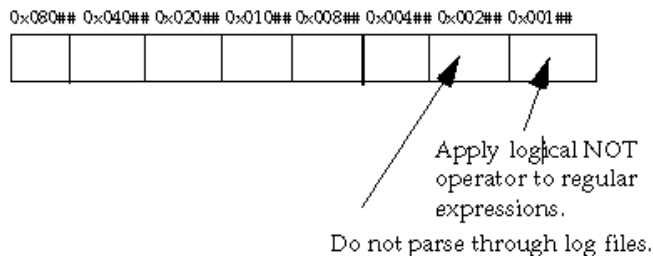


Figure 51. Log Monitor Table-Specific Flag Bits

Log Monitor Table Actions

The SystemEDGE agent provides several default parameters to the action commands when they are invoked. These parameters are in addition to any parameters you specify in the action string and are passed on the command line after those that you

specify. The default parameters are the same as the parameters provided in the SNMP traps sent for the Log Monitor table. Table 53 describes the default parameters for Log Monitor table actions.

Table 53. Parameters for Log Monitor Table Actions

Parameter	Description
trapType	Type of trap being sent, such as logMonMatchEvent or logMonNotReadyEvent.
logMonitorLogFile	NOTE: Name of the file that the agent is monitoring. The file you monitor must be an ASCII-based text file. SystemEDGE does not support monitoring of other character sets, such as Unicode. You can determine a file's encoding by opening it in a text editor and selecting Save As . The encoding is listed in the Save as type field.
logMonitorRegularExpression	Regular expression that the agent is attempting to match for this entry.
logMonitorLastTrap	Time that this trap was sent.
logMonitorLastMatch	Line from the log file that triggered this trap.
logMonitorDescr	Description of this entry.
logMonitorIndex	Index of this entry.
logMonitorFlags	Flags field, in hexadecimal notation (for example, 0x0000), for this entry.

NOTE

The agent logs action-command invocations at the syslog level LOG_DEBUG. It logs action-command invocation errors at syslog level LOG_WARNING. For more information about configuring syslog, refer to Appendix B. For information about starting the agent with its debugging options turned on, refer to Chapter 5, “Starting the SystemEDGE Agent.”

Viewing the Log Monitor Table with AdvantEDGE View

If you are using AdvantEDGE View, you can query a system for Log Monitor table information by selecting the system you want to monitor from the **System** list, selecting **Logfile Monitoring** from the **Configuration** list, and clicking the **Configuration** icon. For more information, refer to the AdvantEDGE View Web Help. Figure 52 shows a sample AdvantEDGE View Log Monitor table.

Index	Description	Logfile	Regexpr	Traps	Last Trap	Last Match	Logfile Size	Logfile Last Update	Action	Flags	Row Status
3	slide attempt	/var/log/daemon-log	slide	1	20 days, 17:02:35	Oct 4 11:09:22 aviewdemo slide[2626]: successful by rdc: slide	191,532	48 days, 23:40:39	(no action)	0x0	●
4	Rlogin attempt	/var/log/syslog	in.rlogin	0	0:00:00	(no match)	219	46 days, 20:31:54	(no action)	0x0	●
5	Telnet attempt	/var/log/syslog	in.telnet	5	46 days, 20:28:01	Oct 30 13:34:58 aviewdemo in.telnetd[5639]: connect from egyptian.empire.com	219	46 days, 20:31:54	(no action)	0x0	●
6	ftp attempt	/var/log/syslog	in.ftp	1	46 days, 20:32:02	Oct 30 13:39:28 aviewdemo in.ftpd[5666]: connect from unixserver	219	46 days, 20:31:54	(no action)	0x0	●
20	Monitor invalid user	/opt/aview/var/log/error_log	user.* not found	142	48 days, 23:40:39	[Thu Nov 1 16:47:30 2001] [error] [client 141.151.44.10] user tech not found: /aviewbin/mainframe.pl	2,988,494	48 days, 23:39:56	(no action)	0x0	●
22	CGI error	/opt/aview/var/log/access_log	cgi-bin/* 40[0-9]	0	0:00:00	(no match)	51,890,984	49 days, 16:56:09	(no action)	0x0	●
1000	WARNING - daemon core	/var/log/syslog	core dumped	0	0:00:00	(no match)	219	46 days, 20:31:54	(no action)	0x0	●
Add Log Monitor Entry											

Figure 52. Sample AdvantEDGE View Log Monitor Table

Configuring the Log Monitor Table

You can control which log files the SystemEDGE agent monitors by adding, deleting, or modifying entries in the Log Monitor table. You can configure the Log Monitor table in the following ways:

- **Dynamically.** Use SNMP commands from a management system, such as AdvantEDGE View, to modify the table. For more information, refer to “Dynamic Configuration During Operation” on page 345.
- **At start-up initialization.** Specify the entries for the Log Monitor table in the SystemEDGE agent configuration file, `sysedge.cf`. For more information, refer to the next section, “Initial Configuration During Start-Up.”

Initial Configuration During Start-Up

On start-up, the SystemEDGE agent reads the `sysedge.cf` configuration file and uses the `watch logfile` directive to specify initial entries to the Log Monitor table. You can add entries to the `sysedge.cf` file to specify the text files that you want the agent to monitor.

Using the watch logfile Directive

You can use the `watch logfile` directive to add entries in the Log Monitor table as follows:

```
watch logfile index flags logFilename 'logMonRegExpr' 'logMonDescr'  
      'logMonAction'
```

Table 54 describes the arguments for the watch logfile directive.

Table 54. Parameters for the watch logfile Directive

Parameter	Description
<i>index</i>	Row number of the entry to be created.
<i>flags</i>	Hexadecimal flags (for example, 0x00001) that direct the additional behavior of this entry.
<i>logFilename</i>	Complete path (starting from root [/]) and file name of the log file to be monitored. NOTE: The file you monitor must be an ASCII-based text file. SystemEDGE does not support monitoring of other character sets, such as Unicode. You can determine a file's encoding by opening it in a text editor and selecting Save As . The encoding is listed in the Save as type field.
' <i>logMonRegExpr</i> '	Regular expression to apply when scanning the log file for matches. You must enclose this value in single quotation marks ('.'). For information about the rules for specifying regular expressions, refer to the UNIX man page on egrep(1).
' <i>logMonDescr</i> '	Description for this entry. It is a quoted string 0 to 128 characters in length and normally contains a description of the file that is being monitored, as well as a severity level for this event.
' <i>logMonAction</i> '	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when the regular expression is matched and a trap is sent. If the string is empty or not specified, the agent invokes no action for this entry.

watch logfile Examples

This section provides examples for using the watch logfile directive.

Searching for pop Connection Attempts

The following example instructs the SystemEDGE agent to add a new entry to the Log Monitor table at table index 15 to search for pop connection attempts on a system:

```
watch logfile 15 0x00 /var/log/syslog 'popper' 'NOTICE - pop connection' ''
```


Searching for su Attempts

The following example instructs the SystemEDGE agent to add a new entry to the Log Monitor table at table index 16 to search for su attempts on a system:

```
watch logfile 16 0x02 /var/adm/messages 'su.*fail' 'WARNING - su attempt'
'/local/bin/mail2admin'
```

The 0x02 flag specifies that the agent should not send traps. Instead, the agent invokes the specified action command.

Dynamic Configuration During Operation

The agent uses the SNMPv2 SMI Row Status textual convention for creating, deleting, and modifying rows in the table. For more information about this convention, refer to Appendix E.

You can use your management system to issue SNMP Set request messages to the SystemEDGE agent in order to modify the entries in the Log Monitor table. Log Monitor table entries are saved to the SystemEDGE agent's `sysedge.mon` configuration file. This ensures that any changes made during the operation of the agent are preserved across agent and system restarts.

NOTE

Configuration file directives in `sysedge.cf` take precedence over entries in `sysedge.mon`. For example, if a Log Monitor table entry is contained in `sysedge.mon` at index 10, and a configuration file directive is added to `sysedge.cf` for index 10 of the Log Monitor table, the entry defined in `sysedge.cf` replaces the entry taken from `sysedge.mon`.

Using the edgewatch Utility to Monitor Log Files

To facilitate log file monitoring, the SystemEDGE agent distribution includes the `edgewatch` command-line utility. `edgewatch` acts in a manager role to configure entries in the Log Monitor table and list entries that currently exist in the table. The `edgewatch` utility is located in the `/bin` subdirectory of the SystemEDGE agent distribution.

You can use edgwatch to add, delete, set the status of, or list entries in the Log Monitor table as follows:

```
edgwatch hostname[:port][,timeout] community logfile command
```

Table 55 describes the parameters of the edgwatch utility.

Table 55. edgwatch Utility Parameters

Parameter	Description
<i>hostname[:port][,timeout]</i>	Specifies the hostname or IP address (in dotted notation) of the system on which the agent exists. If the agent is running on an alternative UDP port (for example, 1691), you must specify that port number through a colon-separator. In addition, you can specify an optional SNMP timeout value (in seconds) using a command separator.
<i>community</i>	Indicates the particular SNMP community string to use when performing the specified command. The community string should have read-write access on the target agent for add, delete and setstatus operations. For the list operation, the community string needs only read access.
<i>command</i>	Specifies the command and associated arguments. Supported commands include add, setstatus, delete, and list. For more information about the commands, refer to the next section, “edgwatch Commands for Log File Monitoring.”

edgwatch Commands for Log File Monitoring

You can use the edgwatch command for log file monitoring as follows:

```
add logMonIndex logMonFlags logFilename "logMonRegExpr" "logMonDescr"
"logMonAction"
setstatus logMonIndex status
delete logMonIndex
list
```

Table 56 describes the arguments for the edgewatch monitoring commands.

Table 56. edgewatch Command Arguments (Page 1 of 2)

Argument	Description
<i>logMonIndex</i>	<p>Specifies the row in the table. Rows are indexed starting at 1. An index of 0 is not permitted.</p> <p>Because SNMP does not include a Create PDU type, new table entries are created as a side effect of setting the columnar values for a non-existent row. Therefore, you must include this value for add operations in order to specify the table index (of an unused row) to use for row creation.</p>
<i>LogMonFlags</i>	Specifies the hexadecimal flags (for example, 0x00001) that directs the additional behavior of this entry.
<i>LogFilename</i>	<p>Complete path (starting from root [/]) and file name of the log file to be monitored. On Windows systems, the log file name must start with a drive letter and absolute path.</p> <p>NOTE: The file you monitor must be an ASCII-based text file. SystemEDGE does not support monitoring of other character sets, such as Unicode. You can determine a file's encoding by opening it in a text editor and selecting Save As. The encoding is listed in the Save as type field.</p>
<i>"logMonRegExpr"</i>	Specifies the regular expression to apply when scanning the log file for matches. You must enclose values for logMonRegExpr in quotation marks (".."). For information about the rules for specifying regular expressions, refer to the UNIX man page on egrep(1).
<i>"logMonDescr"</i>	Specifies the description for this entry. It is a quoted string that is 0 to 28 characters in length and that normally contains a description of the file that is being monitored, as well as a severity level for this event.

Table 56. edgwatch Command Arguments (Page 2 of 2)

Argument	Description
<i>"logMonAction"</i>	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when the agents finds a match for the regular expression and sends a trap. If the string is empty or not specified, the agent invokes no action for this entry.
<i>Status</i>	Used with the setstatus operation, this value specifies the RowStatus textual convention value to use in setting the status of a row in the Log Monitor table. The Row Status parameter is an integer that can take on one of the following values: <ul style="list-style-type: none"> • active(1) • notInService(2) • destroy(6) For more information about the Row Status textual convention, refer to Appendix E.

Sample Uses of the edgwatch Utility

This section provides examples for using the edgwatch utility.

Listing Entries in Log Monitor Table

The following example lists the contents of the SystemEDGE agent's Log Monitor table:

```
edgwatch 127.0.0.1 public logfile list
```

Adding a Log Monitor Entry

The following example instructs the SystemEDGE agent to add a new entry to the Log Monitor table at table index 5 to search for su failures on an HP-UX system. The agent runs the script /local/bin/mail2admin when it finds a match.

```
edgwatch 127.0.0.1 private logfile add 5 0x00 /usr/adm/sulog "SU.* -" "su
attempt - WARNING" "/local/bin/mail2admin"
```

Deleting a Log Monitor Entry

The following example deletes an entry from an agent's Log Monitor table at table index 5:

```
edgewidth 127.0.0.1 private logfile delete 5
```

Disabling a Log Monitor Entry

The following example disables the Log Monitor table entry at table index 5 by setting that entry's status to `notInService(2)`. The entry will remain in the table, but the agent will not scan the log file for the regular expression unless the entry's status returns to `active(1)`.

```
edgewidth 127.0.0.1 private logfile setstatus 5 notInService
```

Removing Log Monitoring Entries

To stop the self-monitoring of a log file, you must remove the appropriate entry from the Log Monitor table. Log Monitor table entries are stored in the `sysedge.mon` file, which ensures that they will not be lost when the SystemEDGE agent restarts. The watch logfile directives in the `sysedge.cf` file create a Log Monitor entry whenever the SystemEDGE agent starts. This row creation results in a new Log Monitor table entry that will be stored in `sysedge.mon`. Thus, permanent removal of a Log Monitor table entry requires two steps:

1. Remove the entry from the `sysedge.cf` file.
2. Remove the entry from the Log Monitor table.

Removing Entries from the `sysedge.cf` File

If you configured a Log Monitor table entry by adding a watch logfile directive to the `sysedge.cf` file, you must manually delete it from that file as part of removing the entry from the table. If you do not remove the `sysedge.cf` directive, the entry will be recreated the next time the SystemEDGE agent starts.

Removing Entries with the edgemon Utility

To remove a log file monitoring entry from the Log Monitor table, use the edgemon utility to delete the entry. The following example deletes row 14 from the Log Monitor table on system 143.45.0.12. Once deleted, the row will be removed both from memory and from the sysedge.mon file.

```
edgemon 143.45.0.12 private logfile delete 14
```

Removing Entries Manually

In some cases, it may not be possible to use the edgemon utility to delete Log Monitor table entries. For example, if you have configured the SystemEDGE agent to disallow SNMP Set operations, the edgemon utility will not work. In this case, you will need to remove the entry from the Log Monitor table by editing the sysedge.mon file and removing the entry from the file. Because this is an active file, you must stop the SystemEDGE agent before you edit it. For more information on the format of this file, refer to Appendix C.

For example, to delete row 14:

1. Stop the SystemEDGE agent.
2. Open sysedge.mon for editing, delete the entry for logmon row 14, and save the file.
3. Open sysedge.cf for editing, delete the entry for logmon row 14 if it exists, and save the file.
4. Restart the SystemEDGE agent.

Recommendations for Log File Monitoring

You can monitor system and application logs to obtain in-depth information about user, system, and application behavior. The following tables provide recommendations for which log files you can monitor and the regular expressions for which you can search for monitoring security, device failures, system capacity, Windows security, and applications and systems.

NOTE

The log files that are described in this section are not the same for all operating systems. These examples are provided for reference; you may need to alter them for your operating system.

Table 57 provides recommendations for files you can monitor for security.

Table 57. Monitoring Security

Description	Log File to Monitor	Regular Expression
WARNING - daemon core	/var/log/daemon-log	core dumped
WARNING - daemon core	/var/log/syslog	core dumped
Monitor SU attempts	/var/adm/messages	su.*fail
Monitor rlogins	/var/log/syslog	in.rlogin
Monitor telnets	/var/log/syslog	in.telnet
Monitor rsh	/var/log/syslog	in.rsh
WARNING - Illegal Instruction, Daemon	/var/log/daemon-log	.*llegal.*nstruction
Spam Relay Attempt	/var/log/syslog	Relaying denied
Monitor DENY packets from a Linux firewall	/var/log/messages	DENY

Table 58 provides recommendations for regular expressions for which you can search in /var/adm/messages to monitor for device failures.

Table 58. Monitoring Device Failures

Description	Regular Expression
Critical: Badtrap Error	. *BAD TRAP.*
Error: SCSI error	. *SCSI.*[E,e]rror.*
Error: SCSI error	. *SCSI.*failed.*
Error: SCSI error	. *SCSI.*hung.*
Critical: badsimms error	. *SIMM.*
Critical: badsimms error	. *BAD.*SIMM.*
Critical: memory error	. *[M,m]emory [E,e]rror.*
Error: disk error	. *disk not responding.*
Error: disk error	. *[D,d]isk.*[E,e]rror.*
'Warning: disk fragmentation error'	. *optimization changed.*
Error: disk error	. *corrupt label.*
Error: I/O error	. *I/O.*[E,e]rror.*
Error: disk read/write errors	. *Error for Command:.*[read,write].*
Error: media error	. *Media Error.*
Info: serialport error	. *zs[0,1,2]: silo overflow.*
Warning: carrier error	. *no carrier.*
Warning: link is down	. *Link Down - cable problem?.*
Error: SDS error	. *[NOTICE,WARNING,PANIC]: md:.*

Table 59 provides recommendations for regular expressions for which you can search in `/var/adm/messages` to monitor system capacity.

Table 59. Monitoring System Capacity

Description	Regular Expression
Critical: memory error	*[O,o]ut of [M,m]emory.*
Critical: memory error	.*[F,f]ile system full.*
Error: diskspace error	.*No space left on device.*

Table 60 provides recommendations for which logs and expressions you can monitor in the Windows event logs to monitor Windows security.

Table 60. Monitoring Windows Security

Description	Event Log	Security Type	Regular Expression
Random Password Hack	Security	All	.*bad
Misuse of Privileges	Security	All	.*[user rights,group management,security change, restart,shutdown]
Improper File Access	Security	Failure	.*[read,write]
Improper Printer Access	Security	Failure	.*print
Virus Outbreak Warning: program files updated	Security	All	.*write.*[exe,dll,com]
Security Policies Change	Application	Information	.*[S,s]ecurity policy

Table 61 provides recommendations for which logs and expressions you can monitor in the Windows event logs to monitor applications and systems.

Table 61. Monitoring Applications and Systems

Description	Event Log	Security Type	Regular Expression
Application Error or Failure	Application	All	.*[F,f]ail.*[E,e]rror
Application Load Problems	Application	All	.*[L,l]oad.*[P,p]roblem
New Software Installed	Application	All	.*[I,N,S,T,A,L,L,I,n,s,t,a,l,l]
Server Process failed during Initialization	Application	All	.*4131
Disk Failures and errors	All	All	.*[D,d]isk
Network Adapter Errors	All	Error	.*[N,n]etwork [A,a]dapter

Monitoring Log File Size

To monitor the size of a log file, add a log file monitoring entry to `sysedge.cf` for an artificial expression, and then set up threshold monitoring for the `logFileSize.x` variable, where `x` is the artificial expression.

For example, do the following:

1. Create a log file monitoring entry as follows:

```
logmon {
  10
  "Monitor Log File Size"
  "/var/log/messages"
  "dontfindanything"
  ""
  0xa
  active
}
```

This example is for a Linux system. For Solaris, specify `/var/adm/messages`, and for HP-UX, specify `/var/adm/syslog/syslog.log`.

2. Create a threshold-monitoring entry that monitors the tenth entry in the Log Monitor table (logMonitorLogFileSize.10, or 1.3.6.1.4.1.546.11.1.8.10) and sends a trap when the log file is larger than 1048576:

```
monentry {
  20
  60
  absoluteValue
  1.3.6.1.4.1.546.11.1.8.10
  gt
  1048576
  active
  0x8
}
```

This example is for a Linux system. For Solaris, specify /var/adm/messages, and for HP-UX, specify /var/adm/syslog/syslog.log.

Rotating Log Files

When you are using log files with rotating (non-static) names, use symbolic links to point a static file name to the current file name. Use scheduled scripts to update the links or the SystemEDGE log file monitoring entries to match the current log file name.

Configuring Windows Event Monitoring

This chapter explains how to use the SystemEDGE agent to monitor Windows event logs for regular expressions.

NOTE

This chapter is relevant only for Windows versions of the SystemEDGE agent.

Monitoring Windows Events

The SystemEDGE agent enables you to instruct the agent to continuously monitor Windows event logs. This Windows event-log monitoring is similar to the standard log file monitoring that is described in Chapter 13, “Configuring Log File Monitoring.”

Whenever a matching event is generated on a system that the agent is monitoring, the SystemEDGE agent notifies the management system with a trap message. It can also execute action commands to handle the event immediately. Because Windows events include several identifying characteristics in addition to the text message, this monitoring capability is somewhat more sophisticated than the standard log file monitoring in the types of matches that you can specify.

Monitoring Windows Event Logs

The flexible NT Event Monitor Table of the Systems Management MIB is located under the nt system branch in the MIB. It enables you to configure the SystemEDGE agent dynamically to monitor event logs for the regular expression that you specify. Each entry in the NT Event Monitor table represents the monitoring of one event log for a particular regular expression.

The NT Event Monitor table monitors the messages that applications log to the Windows event mechanism. Windows event monitoring is also useful in security management; for example, you can configure the agent to monitor the Security Event Log for invalid logins.

Checking Log File Status

When the SystemEDGE agent starts (or after the addition of rows to the NT Event Monitor table), it checks the status of (stats) each Windows event log for its current length and the time that it was last updated. Thereafter, the SystemEDGE agent periodically stats each event log for additions or modifications since the last update. If the event log file has changed, the agent scans only the changes—not the entire event log—to see if a match exists for the specified filters.

If the agent finds a match, it sends an enterprise-specific SNMP `ntEventMonMatch` trap message to the configured management systems. For more information about the `ntEventMonMatch` trap, refer to “`ntEventMonMatch` Trap” on page 224.

Search Criteria

Each Windows Event Monitor table entry instructs the agent to search for matches based on the criteria that are described in Table 62.

Table 62. Criteria for Windows Event Log Searches

Criteria	Description
Event Log	<p>Name of the event log. This value can be any of the following:</p> <ul style="list-style-type: none"> • Application • System • Security • DirService (for Directory Service) • DnsServer (for DNS Service) • FileRepService (for File Replication Service)
Event Type	<p>Type of event. Types 1 through 5 are defined by Windows as the following:</p> <ul style="list-style-type: none"> • error(1) • warning(2) • information(3) • success(4) • failure(5) <p>Type all(6) indicates that the agent should match all event types.</p>
Event Source	<p>Source of the event. This is a text field that contain the name of the program or module that generated the event. The agent uses regular expressions to match this field.</p>
Event Description	<p>Description of the event. This is a text field that describes the event. The agent uses regular expressions to match this field.</p>

The SystemEDGE agent generates an SNMP trap message when it finds a match based on all four criteria. This matching is similar to a Boolean AND operation.

NT Event Monitor Table

For each entry in the NT Event Monitor table, the table provides information such as the following:

- Event log that the agent is monitoring
- Regular expression for which the log is being monitored
- Number of times that a trap has been sent because a match was found
- Time at which the last trap was sent
- Log entry that caused the last match

Columns of the NT Event Monitor Table

Table 63 describes the columns of the NT Event Monitor table. For more information about the NT Event Monitor table, refer to the Systems Management MIB (empire.asn1 in the /doc subdirectory of the SystemEDGE agent distribution).

Table 63. Columns of the NT Event Monitor Table (Page 1 of 3)

Column Name	Permissions	Description
ntEventMonIndex	Read-Only	Row of the table in which this entry exists.
ntEventMonLog	Read-Write	Integer that designates which event log to monitor. The following are possible values: <ul style="list-style-type: none">• Application(1)• Security(2)• System(3)• Directory Service(4)• DNS Service(5)• File Replication Service(6)

Table 63. Columns of the NT Event Monitor Table (Page 2 of 3)

Column Name	Permissions	Description
ntEventMonTime	Read-Only	Time, based on sysUpTime, at which the event occurred.
ntEventMonMatches	Read-Only	Number of times that a match occurred for this entry and the agent sent a trap.
ntEventMonTypeLastMatch	Read-Only	<p>Number that identifies the event type of the last event that matched the search criteria. Types 1 through 5 are defined by Windows as the following:</p> <ul style="list-style-type: none"> • error(1) • warning(2) • information(3) • success(4) • failure(5) <p>Type noMatch(6) indicates that there has not yet been a matching event for this monitoring entry.</p>
ntEventMonTypeFilter	Read-Write	<p>Number that identifies the event type to match for this entry. Types 1 through 5 are defined by Windows as the following:</p> <ul style="list-style-type: none"> • error(1) • warning(2) • information(3) • success(4) • failure(5) <p>Type all(6) indicates that the agent should match all event types.</p>
ntEventMonSrcLastMatch	Read-Only	String that identifies the Event Source of the last event log entry that matched this monitor entry. The Event Source is usually the name of the program that generated the event. Each time a match occurs, this variable is updated.
ntEventMonSrcFilter	Read-Write	Regular expression to apply to the Event Source when scanning the events for matches.

Table 63. Columns of the NT Event Monitor Table (Page 3 of 3)

Column Name	Permissions	Description
ntEventMonDescLastMatch	Read-Only	Description string of the last event log entry that matched this monitor entry. Each time a match occurs, this variable is updated.
ntEventMonDescFilter	Read-Write	Regular expression to apply to the Event Description when scanning the events for matches.
ntEventMonStatus	Read-Write	SNMPv2 RowStatus; one of the following: <ul style="list-style-type: none">• active(1)• notInService(2)• notReady(3)
ntEventMonDescr	Read-Write	Quoted string that is 0 to 128 characters in length and that normally contains a description of the events that are being monitored, as well as a severity level for this event.
ntEventMonAction	Read-Write	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when a match is found and a trap is sent. If the string is empty, the agent invokes no action for this entry.
ntEventMonFlags	Read-Write	Unsigned integer flags value indicating additional behavior that this row should follow during the course of its operation. By default, this field is assigned the hexadecimal value 0x00. For more information about this field, refer to “NT Event Monitor Table Flags” on page 363.

Optimizing Row Creation

You can use the following MIB objects with the NT Event Monitor table to optimize row creation.

Table 64. Scalar Objects for Optimizing Row Creation

MIB Object	Permissions	Description
ntEventUnusedIndex	Read-Only	You can use this variable to optimize table-row creation for the NT Event Monitor table. Perform an SNMP Get of this variable to return an unused index number for the NT Event Monitor table.
ntEventMatchDescr	Read-Write	You can use this MIB object with the ntEventMonMatchIndex MIB object to determine the index number that corresponds to a particular entry description. Perform an SNMP Set of this MIB object to cause the agent to search through entries in the NT Event Monitor table and place the index value of the last entry whose description matches in the ntEventMonMatchIndex MIB object.
ntEventMatchIndex	Read-Only	You can use this MIB object with ntEventMonMatchDescr to match a particular entry description with its index number.

NT Event Monitor Table Flags

You can use the ntEventMonFlags column in the NT Event Monitor Table to specify additional behavioral semantics for the corresponding NT Event Monitor table row. By default, the NT Event Monitor table row does the following:

- Attempts to reinitialize itself
- Sends SNMP traps
- Logs events with the syslog utility
- Invokes actions (if they are configured)

You can set different flag bits to alter these defaults. The SystemEDGE agent interprets all flags in hexadecimal (base 16) notation.

Figure 53 shows the composition of the NT Event Monitor table flags field (ntEventmonFlags).

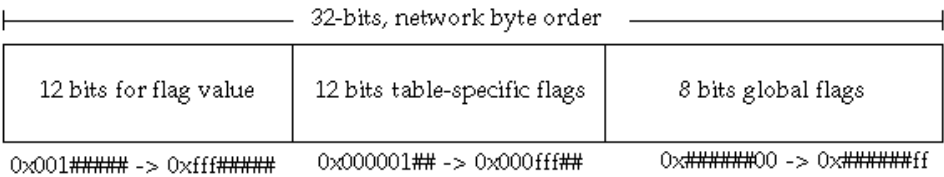


Figure 53. NT Event Monitor Table Flags

The flags value consists of three fields:

- **Field 1:** Common table flags that are defined for all of the self-monitoring tables. This portion is the low-order 8 bits of the flags.
- **Field 2:** Table-specific flags that are defined separately for each self-monitoring table. This field defines the next 12 low-order bits after the common table flags. For Windows event monitoring, there are currently no table-specific flags defined.
- **Field 3:** Flags value reserves the 12 high-order bits for an integer value for use in conjunction with table-specific flags. For Windows event monitoring, there are currently no table-specific flags defined.

NOTE

The following sections define each flag bit. You can combine flag values through logical OR operations.

Table 65 describes the common table flags for the self-monitoring tables.

Table 65. Common Monitor Table Flags (Page 1 of 2)

Flag	Description
0x00000001	Do not execute actions for this entry.
0x00000002	Do not send traps for this entry. This flag bit overrides any other flag bit with respect to traps.
0x00000004	Do not attempt to reinitialize this entry. By default, if the monitored event log is unavailable, the SystemEDGE agent periodically tries to reinitialize this table entry. Setting this bit disables automatic reinitialization.
0x00000008	Do not record events for this entry in the syslog utility. Setting this bit does not affect the sending of traps or execution of actions. On Windows systems, the agent will not log the event to the agent's log file (sysedge.log). When events occur frequently, it is useful to disable event logging.
0x00000010	Send continuous ntEventMonEntryNotReady traps for this entry every time the agent attempts to reinitialize event log monitoring and fails. The agent's default behavior is to send a single ntEventMonEntryNotReady trap when the event log that it is monitoring ceases to exist or when an error accessing that event log occurs. The agent periodically attempts to reinitialize the entry. Enabling this feature causes the agent to send an additional ntEventMonEntryNotReady trap each time reinitialization fails.
0x00000020	Do not pass SystemEDGE arguments to action scripts or programs. SystemEDGE normally passes default action parameters that indicate the trap type, description field, and so on. This flag disables the passing of those arguments. For more information about action parameters, refer to the next section, "NT Event Monitor Table Actions."
0x00000040	Do not send notReady traps for this entry.

Table 65. Common Monitor Table Flags (Page 2 of 2)

Flag	Description
0x00000100	Pre-append the Event ID number to the event description field. The identifier is added using the form [#] where # is the identifier. The purpose of this flag is to facilitate searches for specific Event IDs. For example, to specify that only Event ID 528 should match, specify this flag with the ntEventMonDescFilter set to [528].
0x00000200	Send every event except the specified event. When you enable this flag, the agent applies matches to expressions and does the following: if the match is true, it sets it to false, and if the match is false, it sets it to true. If the results are true, the agent performs the specified action or sends a trap. NOTE: Use caution in setting this flag.
0x####0000	Flag value 0x00010000 → 0xffff0000. The NT Event Monitor Table does not utilize this flag value field; setting it has no effect on the NT Event Monitor Table operation or on any of the supported flag bits.

NT Event Monitor Table Actions

The SystemEDGE agent provides several default parameters to the action commands. These parameters are in addition to any parameters you specify in the action string and are passed on the command line after those that you specify. The default parameters are the same as the parameters provided in the SNMP traps sent for the NT Event Monitor table. Table 66 describes the default parameters for NT Event Monitor table actions.

Table 66. Default Parameters for NT Event Monitor Table Actions (Page 1 of 2)

Parameter	Description
trapType	Type of trap that is being sent, such as ntEventMonMatchEvent or ntEventMonNotReadyEvent.
ntEventMonLog	Event log that the agent is monitoring.

Table 66. Default Parameters for NT Event Monitor Table Actions (Page 2 of 2)

Parameter	Description
ntEventMonTypeLastMatch	Type of event that generated this trap.
ntEventMonTime	Time that this event was generated.
ntEventMonSrcLastMatch	Source of the event that generated this trap.
ntEventMonDescLastMatch	Description of the event that generated this trap.
ntEventMonDescr	Description of this entry.
ntEventMonIndex	Index for this entry.
ntEventMonFlags	Flags field, in hexadecimal notation (for example, 0x0000), for this table entry.

For more information about traps sent by the SystemEDGE agent, refer to Chapter 8, “Private Enterprise Traps.”

NOTE

The SystemEDGE agent logs action-command invocation errors to the sysedge.log file. For information about starting the agent with its debugging options turned on, refer to “Configuring Support for Agent Debugging” on page 135.

Viewing the NT Event Monitor Table with AdvantEDGE View

If you are using AdvantEDGE View, you can query a system for NT Event Monitor table information by selecting the system you want to monitor from the **System** list, selecting **NT Event Monitoring** from the **Configuration** list, and clicking the **Configuration** icon. For more information, refer to the AdvantEDGE View Web Help. Figure 54 shows a sample AdvantEDGE View Monitor table.




Index	Description	Monitor Time	Num Matches	Log	Type Filter	Src Filter	Descr Filter	Last Type Match	Last Src Match	Last Descr Match	Action	Flags	Row Status
30	test	17 days, 5:18:36	124	application (1)	error (1)	*	*	error(1)	Userenv	Windows cannot determine the user or computer name. Return value (1722).	(no action)	0x20	
500	Monitor blah	0:00:00	0	application (1)	error (1)	blah	blah	noMatch (6)	(no match)	(no match)	(no action)	0x20	
12345	eHealth test	0:00:00	0	application (1)	error (1)	*	*	noMatch (6)	(no match)	(no match)	(no action)	0x20	
Add NT Event Monitor Entry													

Figure 54. Sample AdvantEDGE View NT Event Monitor Table

Configuring the NT Event Monitor Table

You can control which event logs the SystemEDGE agent monitors by adding, deleting, or modifying entries in the NT Event Monitor table.

You can configure the NT Event Monitor table in one of these ways:

- **Dynamically.** Use SNMP commands from a management system, such as AdvantEDGE View, to modify the table. For more information, refer to the next section, “Dynamic Configuration During Operation.”
- **At start-up initialization.** Specify the entries for the NT Event Monitor table in the SystemEDGE agent configuration file, `sysedge.cf`. For more information, refer to “Initial Configuration During Start-Up” on page 369.

Dynamic Configuration During Operation

The agent uses the SNMPv2 SMI Row Status textual convention for creating, deleting, and modifying rows in the table. For more information about this convention, refer to Appendix E.

You can use your NMS platform to issue SNMP Set request messages to the SystemEDGE agent in order to modify the entries in the NT Event Monitor Table. NT Event Monitor table entries are saved to the `sysedge.mon` configuration file so that any changes made during the operation of the agent are preserved across agent and system restarts.

NOTE

Configuration file directives in the `sysedge.cf` file take precedence over entries in `sysedge.mon`. For example, if a NT Event Monitor table entry is contained in `sysedge.mon` at index 10, and a configuration file directive is added to `sysedge.cf` for index 10 of the NT Event Monitor table, the entry that is defined in `sysedge.cf` overwrites the entry from `sysedge.mon`.

Initial Configuration During Start-Up

On start-up, the SystemEDGE agent reads the `sysedge.cf` configuration file and uses the `watch ntevent` keyword to specify initial entries to the NT Event Monitor Table. You can specify event logs that you want the agent to monitor by adding appropriate entries for the files, types, and expressions you want to monitor to the `sysedge.cf` file.

Using the watch ntevent Directive

You can use the watch ntevent directive to add entries to the ntEventMonTable as follows:

```
watch ntevent ntEventMonIndex ntEventMonFlags ntEventMonLog
        ntEventMonTypeFilter 'ntEventMonSrcFilter' 'ntEventMonDescFilter'
        'ntEventMonDescr' 'ntEventMonAction'
```

Table 67 describes the parameters of the watch ntevent directive.

Table 67. Parameters for the watch ntevent Directive (Page 1 of 2)

Parameter	Description
<i>ntEventMonIndex</i>	Row number of the entry to be created in the NT Event Monitor Table.
<i>ntEventMonFlags</i>	Hexadecimal flags (for example, 0x00001) that direct the additional behavioral semantics of this entry.
<i>ntEventMonLog</i>	Event log to monitor. One of the following: <ul style="list-style-type: none"> • Application • System • Security
<i>ntEventMonTypeFilter</i>	Event type to match for this entry. The following are valid types: <ul style="list-style-type: none"> • Error • Warning • Information • Success • Failure • All Type All indicates that all event types should match.
<i>'ntEventMonSrcFilt'</i>	Regular expression to apply when scanning the Event Source attribute for each event in the log. You must enclose this value in single quotation marks ('..'). For more information about specifying regular expressions, refer to the UNIX man page on egrep(1).

Table 67. Parameters for the watch ntevent Directive (Page 2 of 2)

Parameter	Description
<code>'ntEventMonDescFilt'</code>	Regular expression to apply when scanning the Event Description attribute for each event in the log. You must enclose this value in single quotation marks ('.'). For more information about specifying regular expressions, refer to the UNIX man page on <code>egrep(1)</code> .
<code>'ntEventMonDescr'</code>	Description for this entry. This value is a quoted string that is 0 to 128 characters in length and that normally contains a description of the file that is being monitored, as well as a severity level for this event.
<code>'ntEventMonAction'</code>	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when the entry is matched and the agent sends a trap. If the string is empty or not specified, the agent invokes no action for this entry.

watch ntevent Directive Examples

This section includes examples for using the watch ntevent directive.

Searching the Application Log for Web Server Messages

The following example adds a new entry to the agent's NT Event Monitor table at table index 1 to search the Application log for messages from the http Web server application:

```
watch ntevent 1 0x00 Application All 'http' '.*' 'Web Server messages' "
```

Searching the Security Log for Failure Events

The following example adds a new entry to the agent's NT Event Monitor table at table index 2 to search the Security log for Failure events that indicate login failures:

```
watch ntevent 2 0x00 Security Failure '.*' '.*' 'Access Failure - WARNING' "
```

Searching the Application Log for Specific Events

The following example adds a new entry to the agent's NT Event Monitor Table at table index 3 to search the Application log for events with Event ID 277:

```
watch ntevent 3 0x0100 Application All '.*' '\[277\]' 'Event ID 277' "
```

The flag 0x0100 adds the Event ID to the description. [277\] is the description field that the agent will attempt to match.

NOTE

The backslash character (\) is required because brackets ([]) are special characters for regular expression matching.

Using the edgewatch Utility to Monitor Windows Events

To facilitate Windows event monitoring, SystemEDGE includes the edgewatch command-line utility. This utility acts in a manager role to configure entries in the NT Event Monitor table and list entries that are currently in the table. The edgewatch utility is located in the /bin subdirectory of the SystemEDGE agent distribution.

NOTE

Although the NT Event monitoring capability is provided on Windows only, you can configure it from any supported platform using the edgewatch utility. That is, you can configure event monitoring on a Windows system by using edgewatch from a Solaris system.

You can use edgewatch to add, delete, set the status of, or list entries in the NT Event Monitor table.

You can use the edgewatch utility as follows:

```
edgewatch hostname[:port][,timeout] community ntevent command
```

Table 68 describes the parameters of the edgewidth utility.

Table 68. Parameters of the edgewidth Utility

Parameter	Description
<i>hostname[:port][:timeout]</i>	Hostname or IP address (in dotted quad notation) of the system on which the SystemEDGE agent exists. If the agent is running on an alternative UDP port (for example, 1691), specify that port number through a colon-separator. In addition, you can specify an optional SNMP timeout value (in seconds) using a command-separator.
<i>community</i>	SNMP community string to use when performing the specified command. The community string should have read/write access on the target agent for add, delete and setstatus operations. For the list operation, the community string need only have read access.
<i>command</i>	Command and associated arguments. Supported commands include add, setstatus, delete, and list. For more information, refer to the next section, “edgewidth Commands for NT Event Monitoring.”

edgewidth Commands for NT Event Monitoring

This section describes the edgewidth commands for the NT Event Monitor table:

```
add ntEventMonIndex ntEventMonFlags EventMonLog ntEventMonTypeFilter
"ntEventMonSrcFilter" "ntEventMonDescFilter" "ntEventMonDescr"
"ntEventMonAction"
setstatus ntEventMonIndex status
delete ntEventMonIndex
list
```

Table 69 describes the parameters of the ntevent commands.

Table 69. Parameters of the ntevent Commands (Page 1 of 2)

Parameter	Description
<i>ntEventMonIndex</i>	Row in the NT Event Monitor table. Rows are indexed starting at 1. An index of 0 is not permitted. Because SNMP does not include a Create PDU type, new table entries are created as side effects of setting the columnar values for non-existent rows. Therefore, this value is required for add operations in order to specify the index (of an unused row) to use for row creation.
<i>ntEventMonFlags</i>	Hexadecimal flags (for example, 0x00001) that direct the additional behavior for this entry.
<i>ntEventMonLog</i>	Event log to monitor. One of the following: <ul style="list-style-type: none"> • Application • System • Security • DirService (for Directory Service) • DnsServer (for DNS Service) • FileRepService (for File Replication Service)
<i>ntEventMonTypeFilter</i>	Event type to match for this entry. The following are valid types: <ul style="list-style-type: none"> • Error • Warning • Information • Success • Failure • All Type All indicates that all event types should match.
<i>"ntEventMonSrcFilt"</i>	Regular expression to apply when scanning the Event Source attribute for each event in the log. You must enclose this value in quotation marks ("..").

Table 69. Parameters of the ntevent Commands (Page 2 of 2)

Parameter	Description
<code>"ntEventMonDescFilt"</code>	Regular expression to apply when scanning the Event Description attribute for each event in the log. You must enclose this value in quotation marks ("..").
<code>"ntEventMonDescr"</code>	Quoted string that is 0 to 128 characters in length and that normally contains a description of the file that is being monitored, as well as a severity level for this event.
<code>"ntEventMonAction"</code>	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when the entry is matched and a trap is sent. If the string is empty or not specified, no action will be taken.
<code>RowStatus</code>	Used with the setstatus operation, this value specifies the RowStatus textual convention value to use in setting the status of a row in the NT Event Monitor table. The RowStatus parameter is an integer that can take on one of the following values: active(1) notInService(2) notReady(3) destroy(6) For more information about the RowStatus convention, refer to Appendix E.

Sample Uses of edgwatch for Monitoring Windows Events

This section provides examples for using the edgwatch utility to monitor Windows events.

Listing Entries in the NT Event Monitor Table

The following example lists the contents of the agent's NT Event Monitor table:

```
edgwatch 127.0.0.1 public ntevent list
```

Adding an NT Event Monitor Entry

The following example adds a new entry to an agent's NT Event Monitor table at table index 5 to search for login failures on a Windows system.

```
edgwatch 127.0.0.1 private ntevent add 5 0x0 Security Failure ".*" ".*"  
"Failed login attempt - WARNING" "\\local\\bin\\mail2admin.exe"
```

This example also instructs the agent to run the `\\local\\bin\\mail2admin.exe` script when the agent finds a match.

Deleting an NT Event Monitor Entry

The following example deletes an entry from an agent's NT Event Monitor table at table index 5:

```
edgwatch 127.0.0.1 private ntevent delete 5
```

Disabling an NT Event Monitor Entry

The following example disables the NT Event Monitor table entry at table index 5 by setting that entry's status to `notInService(2)`. The entry will remain in the table, but the agent will not scan the event log for matches unless the entry's status is returned to `active(1)`:

```
edgwatch 127.0.0.1 private ntevent setstatus 5 2
```

The value 2 corresponds to the Row Status textual convention value `notInService(2)`.

Removing NT Event Monitoring Entries

To stop the self-monitoring of a particular Windows event log, you must remove that entry from the NT Event Monitor table. NT Event Monitor table entries are stored in the `sysedge.mon` file, which ensures that they will not be lost when the SystemEDGE agent restarts. In addition, the `watch ntevent` directives in the `sysedge.cf` file will create an NT Event Monitor

entry whenever the SystemEDGE agent starts. This row creation results in a new NT Event Monitor table entry that will be stored in `sysedge.mon`. Thus, permanent removal of an NT Event Monitor Table entry requires two steps:

1. Remove the entry from the `sysedge.cf` file.
2. Remove the entry from the NT Event Monitor table.

Removing Entries from the `sysedge.cf` File

If you configured an entry by adding a `watch ntevent` directive to the `sysedge.cf` file, you must remove it from that file as part of removing the entry from the table. If you do not remove the `sysedge.cf` directive, the entry will be recreated the next time the SystemEDGE agent is restarted.

Removing Entries with the `edgemon` Utility

To remove an entry from the NT Event Monitor Table, use the `edgemon` utility to delete the entry. The following example deletes row 14 from the NT Event Monitor table on host 143.45.0.12. Once deleted, the row will be removed both from memory and from the `sysedge.mon` file.

```
edgemon 143.45.0.12 private ntevent delete 14
```

Removing Entries Manually

In some cases, you cannot use the `edgemon` utility to delete NT Event Monitor Table entries. For example, if you have configured the SystemEDGE agent to disallow SNMP Set operations, the `edgemon` utility will not work. In this case, you need to remove the entry from the NT Event Monitor Table by editing the `sysedge.mon` file and removing the entry from it. Because this is an active file, you will need to stop the SystemEDGE agent before you edit it. For more information on the format of this file, refer to Appendix C.

For example, to delete row 14:

1. Stop the SystemEDGE agent.
2. Open `sysedge.mon` for editing, delete the entry for `nventmon` row 14, and save the file.
3. Open `sysedge.cf` for editing, delete the entry for `nventmon` row 14 if it exists, and save the file.
4. Restart the SystemEDGE agent.

Configuring History Collection

This chapter describes the SystemEDGE agent's history-sampling capability. It also explains how you can instruct the agent to monitor and store the values of MIB variables over time for future retrieval by a manager.

History Collection

The SystemEDGE agent can track the values of various integer-based MIB objects (counters, gauges, and so on) over time and store them for later retrieval. This functionality, commonly referred to as **history sampling**, can greatly reduce the amount of management station polling across the network. Instead of having to continuously poll to collect the value of a MIB variable over time, the manager can instruct the agent to sample and store the values. The management system can contact the agent periodically to upload the complete history of samples. An additional benefit of history collection is that the agent will continue to sample and store the specified MIB values even during periods of network outage when the management system cannot communicate with the agent.

History Sampling

The agent uses two SNMP MIB tables to provide the history capability:

- **History Control table**, a control table for defining the data-collection functions
- **History table**, a data table for storing the actual data samples

The control table enables you to dynamically configure the agent to sample and store the values of any integer-based MIB variable under its control. The data table stores the values for future retrieval.

To perform baselining and trend analysis, you can configure the SystemEDGE agent to monitor and store the values for swapCapacity, for example, by configuring the agent to sample the value every 5 minutes, and to store the most recent 144 samples. Then, once every 12 hours, the NMS can upload the entire 144 samples to obtain the values for swapCapacity that were collected during the preceding 12-hour period. (The swapCapacity variable of the Systems Management MIB specifies the percentage of swap space that is currently in use).

History Control Table and the Data Table

The History Control table contains parameters that describe the data that the agent will sample and store in the History table. Each row of the History Control table assigns values to the parameters (columnar objects) of the table and thereby defines a specific data-collection function. One or more rows (stored samples) in the History table are associated with that single control row.

Each control table row is assigned a unique value of empireHistoryCtrlIndex. A row defines the data-collection function by specifying the object-instance to be sampled, how often to sample (in multiples of 30 seconds), and the number of samples to keep (**buckets**). Associated with each data-collection

function (row of the control table) is a *set of rows* of the History table. Each row of the History table, which is also called a bucket, holds the value of the specified MIB object that was gathered during one sampling interval.

As each sampling interval occurs, the agent adds a new row to the History table with the same `empireHistoryIndex` as the other rows for this data-collection function. Each new row corresponds to the single row in the History Control table, and has an `empireHistorySampleIndex` that is one greater than the `SampleIndex` of the previous sample.

Columns of the History Control Table

Table 70 describes the columns of the History Control table. For more information about the History Control table, refer to the Systems Management MIB specification (`empire.asn1` in the `/doc` subdirectory of the SystemEDGE agent distribution).

Table 70. Columns of the History Control Table (Page 1 of 2)

Column Name	Description
<code>empireHistoryCtrlIndex</code>	Integer (1 to MAXINT) that uniquely identifies the entry in the table.
<code>empireHistoryCtrlDescr</code>	Text string that describes the data-collection function defined by this entry.
<code>empireHistoryCtrlInterval</code>	Integer value indicating how often (in seconds) the agent should sample the MIB variable. NOTE: The interval must be a multiple of 30 seconds.
<code>empireHistoryCtrlObjID</code>	Complete object-instance identifier of the MIB variable to be sampled. Note that you must include the <i>instance</i> portion of the object identifier (for example, .0 for scalars). The object instance must exist and must be contained within the Systems Management MIB. For example, any supported (INTEGER-based) object in MIB-II, the Host Resources MIB, or the Systems Management MIB is valid. Objects should be of integer type including counter, gauge, integer, or enumerated integer.

Table 70. Columns of the History Control Table (Page 2 of 2)

Column Name	Description
empireHistoryCtrlObjType	ASN1/SNMP type of the MIB variable that the agent is sampling.
empireHistoryCtrlBucketsReq	Requested number of discrete samples to be saved in the History table. Depending on available resources, the agent sets the empireHistoryBucketsGrant column as close to this value as possible.
empireHistoryCtrlBucketsGrant	Actual number of discrete samples that the agent will save in the History table for the data-collection function defined by this entry. The agent will keep the most recent <i>BucketsGrant</i> number of samples.
empireHistoryCtrlLastCall	The last time, based on sysUptime, that a sample was taken on behalf of this entry.
empireHistoryCtrlCreateTime	Time, based on sysUptime, at which this history sampling function was created.
empireHistoryCtrlStatus	<p>One of the following:</p> <ul style="list-style-type: none"> • active • notInService • notReady • createAndGo • createAndWait <p>These values are defined in the SNMPv2 SMI RowStatus textual convention. For more information, refer to Appendix E. Setting the status to destroy(6) causes the agent to discontinue history sampling for this entry, and to delete both this row and the corresponding data sample rows in the History table.</p>

Columns of the History Table

Table 71 describes the columns of the History table. For more information about the data-storage table, refer to the Systems Management MIB specification (*empire.asn1* in the */doc* subdirectory of the SystemEDGE agent distribution).

Table 71. Columns of the History Table

Column Name	Description
empireHistoryIndex	Value that identifies the row in the History Control table of which this sample is a part; that is, it has the same value as the empireHistoryCtrlIndex for the corresponding entry in the control table.
empireHistorySampleIndex	Index that uniquely identifies the particular sample this represents among all samples that are associated with the same history control entry. This index starts at 1 and increases by one as each new sample is stored.
empireHistoryStartTime	Time, based on sysUptime, at which the first sample was taken.
empireHistorySampleTime	Time, based on sysUptime, at which this particular sample was taken.
empireHistoryValue	Current value of the MIB variable taken at this sample.

Optimizing Row Creation

You can use the following scalar MIB objects with the History table to optimize row creation.

Table 72. Scalar Objects for Optimizing Row Creation

MIB Object	Permissions	Description
histCtrlUnusedIndex	Read-Only	You can use this variable to optimize table-row creation for the History table. Perform an SNMP Get of this variable to return an unused index number for the History table.
histCtrlMatchDescr	Read-Write	You can use this MIB object with the histCtrlMatchIndex MIB object to determine the index number that corresponds to a particular entry description. Perform an SNMP Set of this MIB object to cause the agent to search through entries in the History table and place the index value of the last entry whose description matches in the histCtrlMatchIndex MIB object.
histCtrlMatchIndex	Read-Only	You can use this MIB object with histCtrlMatchDescr to match a particular entry description with its index number.

Viewing the History Control Table with AdvantEDGE View

If you are using AdvantEDGE View, you can query a system for History information by selecting the system you want to monitor from the **System** list, selecting **History Collection** from the **Configuration** list, and clicking the **Configuration** icon. For more information, refer to the AdvantEDGE View Web Help. Figure 55 shows a sample AdvantEDGE View History Control table.











Index	Description	Interval (secs)	ObjID	Type	Requested	Granted	Last Call	Create	Row Status
10	 Swap Cap	60	swapCapacity 1.3.6.1.4.1.546.1.1.7.8.30.0	Integer	20	20	49 days, 18:11:06	0:00:03	
11	 test oid	60	totalSwapSpace 1.3.6.1.4.1.546.1.1.7.8.29.0	Integer	60	60	49 days, 18:11:06	0:00:03	
12	 test oid	60	swapCapacity 1.3.6.1.4.1.546.1.1.7.8.30.0	Integer	60	60	49 days, 18:11:06	0:00:03	
15	 Run Queue Length	30	runQLen 1.3.6.1.4.1.546.1.1.7.8.4.0	Gauge	100	100	49 days, 18:11:36	0:00:03	
1000	 swapCapacity history	60	swapCapacity 1.3.6.1.4.1.546.1.1.7.8.30.0	Integer	60	60	49 days, 18:11:06	0:00:03	
<div>Add History Entry</div>									

Figure 55. Sample AdvantEDGE View History Control Table

Configuring the History Control Table

You can control which MIB objects the SystemEDGE agent stores in the History table by adding, deleting, or modifying entries in the History Control table.

You can configure the History Control table in one of these ways:

- **Dynamically.** Use SNMP commands from a management system, such as AdvantEDGE View, to modify the table. For more information, refer to “Dynamic Configuration During Operation” on page 386.
- **At start-up initialization.** Specify the entries for the History Control table in the SystemEDGE agent configuration file, `sysedge.cf`. For more information, refer to the next section, “Initial Configuration During Start-Up.”

Initial Configuration During Start-Up

On startup, the SystemEDGE agent reads the `sysedge.cf` configuration file and uses the `emphistory` directive to specify initial entries to the History Control table. You can specify MIB object for the agent to monitor by adding appropriate entries to `sysedge.cf`.

Using the `emphistory` Command

You can use the `emphistory` keyword to add entries in the History Control table as follows:

```
emphistory index interval objid buckets 'description'
```

Table 73 describes the parameters of the `emphistory` command.

Table 73. `emphistory` Command Parameters

Parameter	Description
<i>index</i>	Row number in which the agent will create the entry.
<i>interval</i>	Interval (in seconds) at which the agent will sample the object's value. The interval must be a multiple of 30 seconds.
<i>objid</i>	Object identifier that specifies the object instance within the agent's MIB whose value should be sampled. The OID can be specified using the complete dotted-decimal value (for example, 1.3.6.1.2.1.25.1.5.0) or the symbolic MIB name (for example, <code>hrSystemNumUsers.0</code>). In both cases, you must specify the object instance, which is normally zero for non-tabular MIB variables.
<i>buckets</i>	Integer number that specifies the number of buckets, or samples, that the agent will store internally. The agent stores the last <i>buckets</i> number of samples. As the agent takes each new sample, it deletes the oldest sample.
<i>'description'</i>	Quoted string (0 to 128 characters in length) that indicates the description field for this entry.

emphistory Command Example

The following entry instructs the agent to sample the value of the object `MemInUse` through table index 15 every 120 seconds and to store the last 60 samples:

```
emphistory 15 120 memInUse.0 60 'MemInUse history'
```

Dynamic Configuration During Operation

You can use the `emphistory` command-line utility to configure entries in the History Control table and to retrieve data samples from the History table dynamically. The `emphistory` utility is located in the `/bin` subdirectory of the SystemEDGE agent distribution. You can also add history table entries to the agent configuration file manually. For more information, refer to the previous section, “Initial Configuration During Start-Up” on page 385.

Using the emphistory Utility

Use the emphistory utility from the command line to add, delete, set the status of, or list entries in the History Control table. You can also use it (with the dump operation) to retrieve stored data samples from the History table.

You can use the emphistory utility as follows:

```
emphistory add ipaddr[:port] commstr ctrlIndex objid interval buckets
    ['description']
emphistory delete ipaddr[:port] commstr ctrlIndex
emphistory list ipaddr[:port] commstr
emphistory setstatus ipaddr[:port] commstr ctrlIndex status
emphistory dump ipaddr[:port] commstr ctrlIndex
```

Table 74 describes the parameters for the emphistory utility.

Table 74. emphistory Utility Parameters (Page 1 of 2)

Parameter	Description
<i>ipaddr[:port]</i>	IP address or hostname of the system on which you want to set the table, optionally followed by the port on which the agent is running. (If the SystemEDGE agent is running on a port other than 161, you must specify a value for <i>port</i> .)
<i>commstr</i>	SNMP community string to use when performing the specified command. To perform add, delete, and setstatus operations, the community string on the target agent must have read-write access. For the list and dump operations, the community string needs only read access.
<i>ctrlIndex</i>	Row number for this entry in the agent's History Control table. Rows are indexed starting at 1. When you specify the dump operation, the agent retrieves all data samples from the History table that correspond to the entry in the History Control table identified by <i>ctrlIndex</i> . If you specify a <i>ctrlIndex</i> value of -1 for dump operations, the agent retrieves <i>all</i> contents of the History table. You must specify a <i>ctrlIndex</i> value for add operations in order to specify the particular MIB table index of an unused row to use for row creation.

Table 74. emphistory Utility Parameters (Page 2 of 2)

Parameter	Description
<i>objid</i>	Complete object-instance identifier of the MIB variable that the agent will sample and store in the History Control table. You can specify the OID using the complete dotted-decimal value (for example, 1.3.6.1.2.1.25.1.5.0) or the symbolic MIB name (for example, hrSystemNumUsers.0). You <i>must</i> provide the instance portion of the OID (that is, .0 for scalars). The object-instance must exist and must be of integer type (which includes counter, gauge, integer, or enumerated integer).
<i>interval</i>	Interval in seconds between successive samples; this value <i>must</i> be a multiple of 30 seconds.
<i>buckets</i>	Number of discrete data samples to be saved in the History table on behalf of this control entry. Each sample, called a bucket, contains the snapshot value of the MIB variable, the time at which the sample was created, the sample index, and an index that corresponds to the entry in the History Control table that defines the data-collection function. The History table stores the most recent <i>buckets</i> number of samples.
<i>status</i>	Used with the setstatus operation, this value specifies the RowStatus textual convention value to use in setting the status of a row in the History Control table. The rowStatus parameter can be one of the following values: <ul style="list-style-type: none"> • active(1) • notInService(2) • destroy(6) Setting the row status to destroy(6) causes the agent to discontinue history sampling for that entry, and to delete the row in the History Control table and the corresponding data sample rows in the History table. For more information about RowStatus textual convention, refer to Appendix E.
<i>'description'</i>	Optional description. You can specify a description when adding a row. If you specify a value, emphistory uses the supplied string instead of the default History Control entry description string. If you include a description, you must enclose it within single quotation marks (' '). The description must be less than 128 characters, not including the quotation marks. Longer strings are truncated.

emphistory Utility Examples

This section provides examples for using the emphistory utility.

Listing Entries in the History Control Table

The following example lists the contents of the agent's History Control table:

```
emphistory list 127.0.0.1 public
```

Adding a History Control Entry

The following example adds a new control entry at table index 5 to the agent's History Control table. This control entry instructs the agent to sample the ifInOctets.1 MIB object-instance every 60 seconds and to store the most recent 10 samples:

```
emphistory add 127.0.0.1 private 5 ifInOctets.1 60 10
```

Deleting a History Control Entry

The following entry deletes the History Control table entry at table index 3:

```
emphistory delete 127.0.0.1 private 3
```

NOTE

This entry also instructs the agent to delete the stored data samples in the History table that correspond to this control entry.

Setting the Row Status of a Control Entry

The following example disables the control entry in the History Control table at table index 5, but it saves the corresponding stored samples in History table:

```
emphistory setstatus 127.0.0.1 private 5 2
```

The value 2 corresponds to the RowStatus textual convention value `notInService(2)`.

Retrieving Stored Data Samples

The following example retrieves all the stored data samples that correspond to the data-collection function defined in row 5 of the History Control table:

```
emphistory dump 127.0.0.1 public 5
```

The following example retrieves *all* the stored samples for all control entries; that is, it retrieves the entire contents of the History table:

```
emphistory dump 127.0.0.1 public -1
```

Adding Custom MIB Objects

The SystemEDGE agent provides a powerful mechanism for extending the Systems Management MIB to include a wide range of information on your systems and applications. Using this feature, you can extend the agent to manage your unique system environment more effectively. You can also design application-specific MIB variables that allow you to manage your applications using SNMP without implementing SNMP support within the application source.

Systems Management MIB Extension Group

The SystemEDGE agent provides this extension support through the Extension group (extensionGroup) of the Systems Management MIB. This group contains unspecified scalar MIB variables that you can configure. In response to a SNMP GetRequest for one of these variables, the SystemEDGE agent invokes the command that you specify for the variable and returns the value that is returned from the command. Using the SNMP Set operation, you can also pass parameters to a command.

The Extension group is supported on both UNIX and Windows systems; on Windows, however, you may also need to configure the agent to report on performance and configuration data that is contained in the Windows registry. To support this reporting, the SystemEDGE agent also provides a Windows registry

extension group. For more information about this group, refer to Chapter 14, “Configuring Windows Event Monitoring.” Figure 56 shows four sample extension variables that are distributed with the agent.

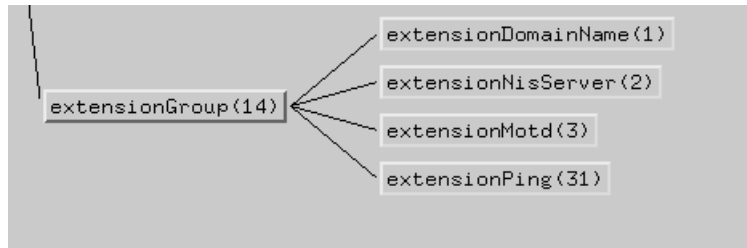


Figure 56. Sample Extension Group

Features of the Extension Group

The Extension Group of the Systems Management MIB is located at OID 1.3.6.1.4.1.546.14. This group provides space for 2^{32} user-defined scalar MIB variables. These new variables are numbered 1 through 2^{32} and are referenced as any other scalar MIB object. For example, extension object 1 is referred to in Get and Set request messages as 1.3.6.1.4.1.546.14.1.0. In all cases, extension variables use the object-instance identifier of .0.

An extension variable can be any valid base SNMP type, including the following:

- Integer
- Counter
- Gauge
- Octetstring
- TimeTicks
- ObjectId
- IpAddress

You can specify variables as Read-Only or Read-Write.

NOTE

The SystemEDGE agent logs extension command invocations at syslog level LOG_DEBUG. It logs extension-command invocation errors at syslog level LOG_WARNING. For more information about configuring the syslog utility, refer to Appendix B. For more information about starting the agent with its debugging options turned on, refer to Chapter 5, “Starting the SystemEDGE Agent.”

Configuring Extension Variables

You can configure extension variables in the SystemEDGE agent configuration file, `sysedge.cf`. On startup, the agent configures these values and verifies whether the program or command associated with the variable is executable. Within the configuration file, the keyword `extension` defines an extension variable. The next section describes how to use this keyword.

Using the extension Keyword

You can use the `extension` keyword to add entries in the extension group as follows:

```
extension LeafNumber Type Access 'Command'
```

Table 75 describes the parameters of the extension command.

Table 75. Parameters for the extension Command

Parameter	Description
LeafNumber	Extension variable number in the range of 1 through 2^{32} that is being defined by this entry.
Type	SNMP type for this entry. The supported types are as follows: <ul style="list-style-type: none">• Integer• Counter• Gauge• Octetstring• TimeTicks• Objectid• IPAddress
Access	One of the following: <ul style="list-style-type: none">• Read-Only• Read-Write
'Command'	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when this variable is accessed through either a Get, GetNext or Set request. If the command file is not currently accessible, the configuration of this variable will fail.

Additional Parameters

In addition to the parameters that you specify in the configuration file as part of the extension command, the SystemEDGE agent passes some additional parameters to help you decide how to treat the request. These parameters are passed after any parameters that you specified in quotation marks as part of the extension statement.

Table 76 describes the additional parameters that can be passed to the extension command.

Table 76. Additional Parameters for the extension Command

Parameter	Description
LeafNumber	An integer value that represents the leaf number of this variable (1 through 2^{32}). If you have a single script that supports multiple extension values you can use this parameter to determine which variable is being requested.
Request-Type	A string that indicates the type of SNMP request: <ul style="list-style-type: none">• Get• GetNext• Set The request type is passed with all letters capitalized.
Set-Value	A string that contains the value that is passed in a Set request. Use this string in your extension program to modify the current value of the variable in such a way that a future Get or GetNext can return this value.

Extension Examples

The SystemEDGE agent includes several sample extension variables. These examples are defined in the sample `sysedge.cf` file. The scripts that implement these examples are included in the `/contrib` subdirectory of the SystemEDGE agent distribution. These variables are also defined in the Systems Management MIB (`empire.asn1` in the `/doc` subdirectory of the SystemEDGE agent distribution).

CAUTION

Before you add your own extensions, carefully review the examples in this chapter and in the Systems Management MIB. For clarity, these examples include the appropriate configuration file extension commands.

You can add these extensions to your `/etc/sysedge.cf` (UNIX) or `%SystemRoot%\System32\sysedge.cf` (Windows) file to make them available to the SystemEDGE agent.

DNS Domain (UNIX Only)

The following extension object returns the DNS domain name of the underlying system (as opposed to the Network Information System [NIS] domain name):

```
extension 1 OctetString ReadOnly /opt/EMPSysedge/contrib/getextension.sh
```

The instance-identifier of this object is 1.3.6.1.4.1.546.14.1.0.

NIS Domain Name (UNIX Only)

The following extension object returns the NIS domain name of the underlying system (as opposed to the DNS domain name):

```
extension 2 OctetString ReadOnly /opt/EMPSysedge/contrib/getextension.sh
```

The instance-identifier of this object is 1.3.6.1.4.1.546.14.2.0.

Remote Pinger (UNIX and Windows)

The following extension objects allow you to instruct the SystemEDGE agent to ping a remote host from the host where the agent is running. This can be a useful tool for diagnosing network connectivity problems. It is also a good example of how to use SNMP Set operations with extension variables.

For UNIX systems, enter the following in the `/etc/sysedge.cf` file:

```
extension 31 OctetString ReadWrite /opt/EMPsysedge/contrib/ping.sh
```

For Windows systems, enter the following in the `%SystemRoot%\System32\sysedge.cf` file:

```
extension 31 OctetString ReadWrite c:\sysedge\contrib\ping.bat
```

The instance-identifier of this object in both examples is 1.3.6.1.4.1.546.14.31.0.

To use this feature, first Set the variable with the name or IP address of the destination you would like to ping. Then, when you Get the variable, the agent returns the output from the ping attempt.

Writing Extension Scripts

The SystemEDGE agent places very few constraints on the operation of extension scripts. It does, however, require two things:

- All operations (Set, Get, or GetNext) must have output. The output of Set invocations should echo the value that was actually Set while the output of Get and GetNext should be the object's value. If there is no output, the SNMP query will fail.
- Output from extension scripts is only parsed up through the first newline character. If a newline is the first character, the output is considered NULL, causing the SNMP query to fail, and returning an error.

NOTE

Because the SystemEDGE agent runs as root or administrator, you must ensure that all commands and scripts use absolute pathnames, are fully debugged, and contain no ambiguous code or unnecessary options.

On Windows systems, you can use batch files for writing extension scripts. The agent can directly execute those batch files. However, batch file functionality is severely limited. Use Perl and other scripting languages for Windows instead.

Testing Your Script: An Example

After you create an extension script, test it at the command line. The following example creates an OctetString extension on a UNIX system, tests its output, and then uses SNMP to return the value from the script:

1. Add the following line to `/etc/sysedge.cf`:

```
extension 1 OctetString ReadWrite /opt/EMPsysedge/debugext.sh
```

NOTE

This example tests an extension script that is an OctetString. It is valid for UNIX operating systems.

On Windows systems, you must call the interpreter in your action script command. For example, enter `perl.exe myscript.pl`.

2. Test the setup by entering the following at the command line:

```
./debugext.sh 1 SET mycommandlinetest
your_output_here
./debugext.sh 1 GET
your_output_here
```

3. Enter the following at the command line to create a file called `myset.txt`:

```
echo "1.3.6.1.4.1.546.14.1.0 04 debugSetString" > myset.txt
```

You now have a file called `myset.txt`. You are setting the `OctetString(04)` “`debugSetString`” to the OID `1.3.6.1.4.1.546.14.1.0`.

4. Issue the SNMP Set by entering the following:

```
./snmpset private 127.0.0.1 < myset.txt
```

5. Retrieve the value of your `debugSetString` by entering the following:

```
./snmpget public 127.0.0.1 1.3.6.1.4.1.546.14.1.0 debugSetString
```

For more information, refer to the `debugext.sh` example that exists on the SystemEDGE Contributed Information page at www.concord.com/sysedge-contrib.

Using Extension Variables with Your Management Software

All methods for incorporating extension variables into your management system software (for example, Cabletron Spectrum, HP OpenView, Sun Enterprise Manager, and so on) require you to edit and import MIB specification files. This guide does not discuss details of importing MIB specification files into management system software, but it does describe the two overall strategies exist for incorporating extension variables:

- Edit `empire.asn1` to include the extension variables that are defined for your site.
- Edit a separate MIB file to include the extension variables that are defined for your site.

Editing `empire.asn1` for Extension Variables

Follow these steps to add your own extension variables to the Systems Management MIB (`empire.asn1`):

1. Create and debug the relevant extension scripts, and then configure them in the agent's configuration file, `sysedge.cf`, to include them.
2. Edit `empire.asn1` to include new extension MIB variables that exist under the `extensionGroup`.
3. Perform a test compile of `empire.asn1` to ensure there are no syntax errors. This procedure is specific to your management system and MIB compiler.
4. Import the new `empire.asn1` file into your management system software. This procedure is specific to your management system and MIB compiler.

NOTE

If you do not reimport the MIB file, your management system software will not be able to access the new extension MIB objects.

Editing a Separate MIB Specification for Extension Variables

You can place extension variables in a separate MIB specification file for ease of updating. In this way, you can save time and effort by making changes to a MIB specification file that is much smaller than `empire.asn1`, instead of recompiling and reloading the entire `empire.asn1` file.

To add your extension variables to a separate MIB specification file:

1. Create and debug the relevant extension scripts, and configure them in the agent's configuration file, `sysedge.cf`, to include them.

2. Edit your own MIB specification file (for example, `extensions.asn1`) to include new extension MIB variables that exist under the Systems Management Extension group.
3. Perform a test compile of `extensions.asn1` to ensure that there are no syntax errors. This procedure is specific to your management system and MIB compiler.
4. Import the `extensions.asn1` file into your NMS software. This procedure is specific to your management system and MIB compiler.

NOTE

If you do not reimport the MIB file, your management system software will not be able to access the new extension MIB objects.

Recommendations for Using Extensions

When you create SystemEDGE extensions, follow these guidelines:

- Do not edit `empire.asn1`. Use a separate MIB specification.
- Use SystemEDGE debugging (log file monitoring) to find output, plug-in, and extension problems.
- Use 128 characters or less for extension entries to meet the Windows limitation on command line length. Remember that the length includes paths, commands, and arguments.
- Do not use batch files. Instead use Windows scripting, VBScript, Perl, C, Shell, and so on.
- Watch the fork/exec time for new extensions. Extensions are synchronous and can block other actions.

Adding Windows Registry and Performance MIB Objects

The SystemEDGE agent provides a powerful mechanism for extending the Systems Management MIB to include information from the Windows registry and performance counters. This information includes configuration data (which is normally viewed with regedit) and performance data (which is normally viewed with perfmon).

Using this feature, you can customize the SystemEDGE agent to return additional configuration and performance data for your systems and applications. For example, you can use the SystemEDGE agent to make many application registry entries that specify the configuration of the application available through SNMP. In addition, you can access performance statistics for many applications through the SystemEDGE agent.

Systems Management MIB ntRegPerf Group

The SystemEDGE agent provides the support for these additional configuration and performance parameters in the ntRegPerf group of the Systems Management MIB. This group contains 128 unspecified scalar MIB variables that you can configure. In response to a SNMP Get request for one of these variables, the SystemEDGE agent will read the Windows registry and return the value obtained.

Figure 57 shows two sample ntRegPerf variables that are distributed with the SystemEDGE agent.

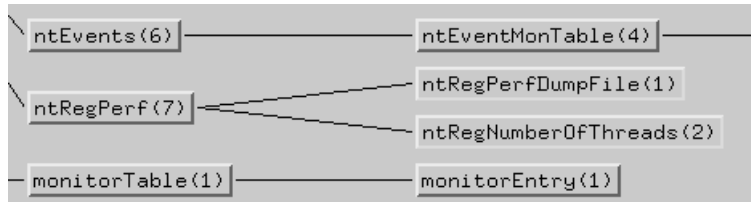


Figure 57. Sample ntRegPerf Group

Windows Registry and Performance Functionality

The ntRegPerf group of the Systems Management MIB is located at OID 1.3.6.1.4.1.546.5.7. This group provides a space for up to 128 user-defined scalar MIB variables. These new variables are numbered 1 through 128 and are referenced just like any other scalar MIB object. For example, ntRegPerf object 1 is referred to in Get request messages as 1.3.6.1.4.1.546.5.7.1.0. In all cases, ntRegPerf variables use the object-instance identifier of .0.

An ntRegPerf variable can be any valid base SNMP type, including the following:

- Integer
- Counter
- Gauge
- OctetString
- TimeTicks
- ObjectId
- IpAddress

Registry Data

The SystemEDGE agent provides data from the standard configuration registry. This data is indexed by both a key name and a value. For example, the key `SYSTEM\CurrentControlSet\Control\CrashControl` and the value `DumpFile` identify a text string that describes the location of the system dump file. Only `LOCAL_MACHINE` registry data is supported. Table 77 matches the registry data types that the SystemEDGE agent supports with the preferred SNMP type.

Table 77. Supported Registry Data Types

Registry Data Type	Suggested SNMP Types
REG_DWORD	Integer
REG_SZ	OctetString
REG_EXPAND_SZ	OctetString
REG_MULT_SZ	OctetString; only the first string is returned

Performance Data

The SystemEDGE agent provides access to performance counters in the performance registry by specifying the object and counter names. For instance, the object `NWLink NetBIOS` and counter `Bytes Total/sec` identifies the NetBIOS byte-counter statistic.

When using performance counters, you must carefully interpret the data. At this time, there are at least 27 different counter types in Windows. Most counters do not actually report the data in the format you would assume based on the name of the counter. Instead, most values are raw counters that require you to do some post-processing, such as dividing the difference of two samples by the elapsed time. If you see a counter name like `Bytes Total/sec`, the data you are really getting is a raw counter that can be used to calculate this rate value.

NOTE

Familiarize yourself with Windows Performance counters before using this feature of the SystemEDGE agent. For more information, refer to the section on optimizing Windows in the Windows Resource Kit.

Table 78 matches the 4-byte performance data types that the SystemEDGE agent supports with the preferred SNMP types.

Table 78. Supported 4-Byte Performance Counter Types

4-Byte Performance Data Type	Suggested SNMP Types
PERF_COUNTER_COUNTER	Counter or integer
PERF_COUNTER_RAWCOUNT	Counter, integer, or gauge
PERF_COUNTER_RAWCOUNT_HEX	Counter, integer, or gauge
PERF_SAMPLE_COUNTER	Counter, integer, or gauge

Table 79 matches the 8-byte performance data types that the SystemEDGE agent supports with the preferred SNMP types. Because SNMPv1 supports only 4-byte values, SystemEDGE will return only the least significant 4-bytes of data.

Table 79. Supported 8-Byte Performance Data Types (Page 1 of 2)

8-Byte Performance Data Type	Suggested SNMP Types
PERF_COUNTER_TIMER	Counter, integer, or gauge
PERF_COUNTER_BULK_COUNT	Counter, integer, or gauge
PERF_COUNTER_LARGE_RAWCOUNT	Counter, integer, or gauge
PERF_COUNTER_LARGE_RAWCOUNT_HEX	Counter, integer, or gauge
PERF_COUNTER_TIMER_INV	Counter, integer, or gauge
PERF_AVERAGE_BULK	Counter, integer, or gauge

Table 79. Supported 8-Byte Performance Data Types (Page 2 of 2)

8-Byte Performance Data Type	Suggested SNMP Types
PERF_100NSEC_TIMER	Counter or integer
PERF_100NSEC_TIMER_INV	Counter or integer
PERF_COUNTER_MULTI_TIMER	Counter or integer
PERF_COUNTER_MULTI_TIMER_INV	Counter or integer
PERF_100NSEC_MULTI_TIMER	Counter or integer
PERF_100NSEC_MULTI_TIMER_INV	Counter, integer, or gauge
PERF_ELAPSED_TIME	Integer
PERF_RAW_FRACTION	Integer or gauge

Unsupported Performance Data Types

Several counters require significant post-processing with multiple samples and internal data that is difficult to present in a single value. For that reason, SystemEDGE does not support the following counter types:

- PERF_COUNTER_QUEUELEN_TYPE
- PERF_COUNTER_TEXT
- PERF_COUNTER_NODATA
- PERF_SAMPLE_BASE
- PERF_AVERAGE_TIMER
- PERF_AVERAGE_BASE
- PERF_COUNTER_MULTI_BASE
- PERF_RAW_BASE
- PERF_COUNTER_HISTOGRAM_TYPE

Configuring Windows Registry and Performance Variables

Windows Registry and Performance variables are configured in the SystemEDGE agent configuration file, `sysedge.cf`. When the SystemEDGE agent starts, it configures these values and verifies that the value associated with each variable is accessible. If not, it prints an error message to `sysedge.log` and does not create the MIB object.

Using the `ntRegPerf` Keyword

Within the configuration file, the keyword `ntregperf` defines the `ntRegPerf` variable. You can use the `ntRegPerf` keyword to add entries in the `ntregperf` group as follows:

```
ntregperf LeafNumber Type Registry 'Key' 'Value'
```

or

```
ntregperf LeafNumber Type Performance 'Object' 'Counter' 'PerfInstance'
```

Table 80 describes the parameters for the `ntRegPerf` command.

Table 80. Parameters for the `ntRegPerf` Command (Page 1 of 2)

Parameter	Description
<i>LeafNumber</i>	<code>ntRegPerf</code> variable number in the range of 1 through 128 that is being defined by this entry.
<i>Type</i>	SNMP type for this entry. The supported types are as follows: <ul style="list-style-type: none">• Integer• Counter• Gauge• Octetstring• TimeTicks• Objectid• IPAddress

Table 80. Parameters for the ntRegPerf Command (Page 2 of 2)

Parameter	Description
<i>Registry</i>	Selects a configuration registry entry.
<i>Performance</i>	Selects a configuration registry entry.
<i>'Key'</i>	Quoted string that is 0 through 512 characters in length and that specifies the registry key to be accessed.
<i>'Value'</i>	Quoted string that is 0 through 128 characters in length and that specifies the registry key's value to be accessed.
<i>'Object'</i>	Quoted string that is 0 through 512 characters in length and that specifies the performance object to be accessed.
<i>'Counter'</i>	Quoted string that is 0 through 128 characters in length and that specifies the object's performance counter value to be accessed.
<i>'PerfInstance'</i>	Quoted string that specifies the performance counter instance to be accessed; it should be equivalent to that listed in the Windows perfmon utility.

Windows Registry and Performance Examples

The SystemEDGE agent includes several sample ntRegPerf variables. These examples are defined in the sample `sysedge.cf` file. Before you add your own ntRegPerf extension, study these examples and their definitions in the Systems Management MIB (`empire.asn1` in the `/doc` subdirectory of the SystemEDGE agent distribution). For clarity, these examples include the appropriate configuration file ntRegPerf commands.

CrashControl DumpFile

The following ntRegPerf object returns the path to the dump file:

```
ntregperf 1 OctetString Registry
  'SYSTEM\CurrentControlSet\Control\CrashControl' 'DumpFile'
```

The object instance-identifier of this object is 1.3.6.1.4.1.546.5.7.1.0.

Objects Threads

The following ntRegPerf object returns the total number of threads that are currently in the system:

```
ntregperf 2 Gauge Performance 'Objects' 'Threads' '1'
```

The object instance-identifier of this object is 1.3.6.1.4.1.546.5.7.2.0.

TCP Segments Sent/Sec

The following ntRegPerf object returns the total number of TCP segments that were transmitted by the system:

```
ntregperf 3 Counter Performance 'TCP' 'Segments Sent/sec' '1'
```

The object instance-identifier of this object is 1.3.6.1.4.1.546.5.7.3.0.

Using Windows Registry and Performance Variables with Your Management Software

There are several methods for incorporating ntRegPerf variables into your management system software (for example, Aprisma Spectrum, HP OpenView, Sun Enterprise Manager, and so on), all of which require editing and importing MIB specification files. While the details of importing MIB specification files into management system software are beyond the scope of this guide, two overall strategies exist for incorporating ntRegPerf variables:

- Edit empire.asn1 to include the ntRegPerf variables that are defined for your site.
- Edit a separate MIB file to include the ntRegPerf variables that are defined for your site.

Editing empire.asn1 for ntRegPerf Variables

Follow these steps to add your own ntRegPerf variables to the Systems Management MIB:

1. Create and debug the relevant ntRegPerf entries in the agent's configuration file, sysedge.cf, to include them.
2. Edit empire.asn1 to include new ntRegPerf MIB variables.
3. Perform a test compile of empire.asn1 to ensure there are no syntax errors. This procedure is specific to your management system and its corresponding MIB compiler.
4. Import the new empire.asn1 file into your management system software. This procedure is specific to your management system and MIB compiler.

NOTE

If you do not reimport the MIB file, your management system software will not be able to access the new MIB objects.

Editing a Separate MIB Specification for ntRegPerf Variables

You can place ntRegPerf variables in a separate MIB specification file for ease of updating. In this way, you can save time and effort by making changes to a MIB specification file that is much smaller than empire.asn1, instead of recompiling and reloading the entire empire.asn1 file.

1. Create and debug the relevant ntRegPerf entries in the agent's configuration file, sysedge.cf, to include them.
2. Edit your own MIB specification file (for example, ntregperf.asn1) to include the new ntRegPerf MIB variables that exist under the Systems Management MIB ntregperfGroup.

3. Perform a test compile of the ntregperf.asn1 file to ensure that there are no syntax errors. This procedure is specific to your management system and its corresponding MIB compiler.
4. Import the ntregperf.asn1 file into your management system software. This procedure is specific to your network management station and its corresponding MIB compiler.

NOTE

If you do not reimport the MIB file, your management system software will not be able to access the new MIB objects.

Deploying the SystemEDGE Agent

This chapter describes deployment options for the SystemEDGE agent.

Introduction

Deploying the SystemEDGE agents can be challenging in large distributed environments. This chapter provides guidelines and suggestions for automating the deployment of the SystemEDGE agent.

For small numbers of systems, deployment by hand may be advantageous because it requires little configuration or preparation. However, as the number of systems and locations grows, the effort that is required to manually deploy new software grows exponentially.

SystemEDGE includes a sample set of scripts that you can use to help automate its deployment. The scripts are contained in the /contrib subdirectory of the SystemEDGE agent distribution under auto.install. The /contrib subdirectory also includes an ntdist.pl script, which you can download from the SystemEDGE Contributed Information page. You can run this script from the command line on Windows systems, using Perl with the Win32 extension. For complete instructions, refer to <http://www.concord.com/sysedge-contrib>.

SystemEDGE also provides the following deployment options:

- Using AdvantEDGE View Agent Deployment
- Downloading the agent from a Web page
- Distributing the agent through e-mail

Deploying SystemEDGE with AdvantEDGE View

If you are using SystemEDGE with AdvantEDGE View on Windows systems, you can automatically deploy SystemEDGE agents and eHealth AIMs through AdvantEDGE View Agent Deployment.

To deploy SystemEDGE and eHealth AIMs from AdvantEDGE View:



1. Click the **Administration** icon. AdvantEDGE View displays the Administration page.



2. Click the **Agent Deployment** icon. AdvantEDGE View displays the AdvantEDGE View: Agent Deployment form. For details about completing this form, refer to the *eHealth AdvantEDGE View User Guide* or the AdvantEDGE View Web Help.

How the Automated Deployment Works

AdvantEDGE View obtains a list of target systems to deploy, including information on deployment options for each system, checks for local files, and verifies that the correct files for deployment exist on the system from which you will deploy the software. If the deployment setup is invalid, AdvantEDGE View stops the deployment.

AdvantEDGE View next checks each target system to ensure that it meets the deployment criteria. For more information about the criteria, refer to the *eHealth AdvantEDGE View User Guide*. If the target system meets the installation criteria, AdvantEDGE View copies the installation files (based on the options you specified on the Agent Deployment form) to the

target systems. If the system does not meet the installation criteria, AdvantEDGE View displays an error on the Deployment Results Summary, and moves on to the next target system.

Deploying SystemEDGE from the Web

You can also deploy the SystemEDGE agent, eHealth AIMs, and other modules to your systems from a Web page within AdvantEDGE View or from the eHealth Software Downloads Web page.

To access the AdvantEDGE View download page:



1. Click the **Agent Deployment** icon. AdvantEDGE View displays the SystemEDGE Deployment form.
2. Click the link in the **Agent Downloads** section of the form. The Agent Downloads page appears, displaying the products that are available for deployment.
3. Click the **README** link next to the package you want to install for installation instructions and the **Package** link to begin downloading the installation package.
4. Click **Latest downloads from Concord** to display the Software Downloads page of the eHealth Support Web site. For more information, refer to the next procedure.

To access the Software Downloads page of the eHealth Support Web site:

1. In your Web browser, go to <http://support.concord.com>.
2. Enter your user name and password, then click **Login**.
3. Select **downloads**. The Software Downloads page appears.
4. Click the product that you want to download. A Web page appears, listing the available versions of that product.
5. Click the **Instructions** link for installation instructions and then the **Software Package** link to begin downloading the SystemEDGE agent or eHealth AIMs to your system.

Deploying SystemEDGE through E-mail

You can deploy the SystemEDGE agent, eHealth AIMs, and other modules to your users through e-mail. To do so, copy the installation package file, the readme.txt file, and the user guide from your AdvantEDGE CD-ROM, and e-mail them to the user. You can modify the readme.txt file as necessary for your deployment.

Third-Party Deployment Tools

For new systems, most vendors provide automated installation tools that can install third-party software at operating system installation time. Table 81 lists some recommended programs for automating operating system installation.

Table 81. Recommended Automated Operating System Installation Programs

Program	URL
Sun JumpStart (Migration Kit)	http://www.sun.com
HP™ Ignite-UX	http://www.hp.com/
Microsoft System Management Server (SMS)	http://www.microsoft.com/
Symantec™ Ghost®	http://www.symantec.com/
PowerQuest™ Drive Image®	http://www.powerquest.com/
Red Hat Kick Start	http://www.redhat.com/

Table 82 lists some recommended software deployment tools.

Table 82. Recommended Software Deployment Tools

Deployment Tool	URL
Tivoli®	http://www.tivoli.com
Computer Associates™ (CA)	http://www.ca.com
Microsoft System Management Server (SMS)	http://www.microsoft.com
HPOV™ Software Distributor	http://www.hp.com/openview/products/softdist.html

Even if you choose not to implement a software distribution system, you can take some steps to help automate deployment of the SystemEDGE agent in a distributed environment.

Automating Deployment

Automating software deployment involves four steps:

1. Making software available to remote systems
2. Installing software on remote systems
3. Configuring software for distributed systems
4. Licensing the software remotely

Making Software Available to Remote Systems

You can make software available to remote systems in a variety of ways, including through protocols, remote file systems, and e-mail.

Using Protocols to Distribute Software

This section discusses the protocols that you can use to distribute software.

Hypertext Transfer Protocol (HTTP). HTTP is the set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web (WWW). Relative to the TCP/IP suite of protocols (which are the basis for information exchange on the Internet), HTTP is an application protocol. You can easily distribute SystemEDGE agents through the Web by placing them on a Web page for downloading and installation.

File Transfer Protocol (FTP). FTP, a standard Internet protocol, is the simplest way to exchange files between computers on the Internet. Like HTTP, which transfers Web pages and related files, and SMTP, which transfers e-mail, FTP is an application protocol that uses TCP/IP. FTP is commonly used to transfer Web page files from the computer where they were created to the computer that acts as their server. FTP is also commonly used for downloading programs and other files to your computer from other servers.

Remote Copy Protocol (RCP). RCP enables you to integrate remote copy operations into your applications. An application can copy files between the local and remote machine, or from one remote machine to another. RCP commands support recursive file copying and can preserve the original time and date attributes of the file.

Remote Distribution (RDIST). RDIST maintains identical copies of files over multiple hosts. It preserves the owner, group, mode, and modification time of files and can update programs that are executing. Almost all versions of UNIX include RDIST, but most include a very old version, sometimes referred to as 4.2BSD rdist, rdist classic, or rdist version 3.

Network File System (NFS). NFS is a client/server file-sharing protocol that lets you view and optionally store and update files on a remote computer as though they were on your own computer. To use NFS, your system must have an NFS client, and the other computer must have the NFS server. Both systems must also have TCP/UDP/IP installed.

Microsoft Windows for Workgroups, Windows 95, and Windows NT include client and server SMB protocol support.

Server Message Block Protocol (SMB). SMB provides a method for client applications on one computer to read and write to files on, and to request services from, server programs in a computer network. You can use SMB over the Internet on top of its TCP/IP protocol, or on top of other network protocols, such as IPX and NetBEUI. Using the SMB protocol, an application (or the user of an application) can access files at a remote server as well as other resources, including printers, mailslots, and named pipe. Thus, a client application can read, create, and update files on the remote server. It can also communicate with any server program that is set up to receive an SMB client request.

Installing Software on Remote Systems

You can install software on remote systems in several ways. Using a third party deployment system is recommended for a large preinstalled base. You can also use the operating system-specific installation packages that are included with the distribution CD and copy the required files.

Configuring Software for Distributed Systems

You can configure software for a larger distributed system in several ways, but in all cases, it is desirable to accomplish this remotely. You can generate a cookie-cutter configuration file that can be used by all systems or classes of systems (for example, all Windows NT 4.0 machines or all UltraSPARC systems), and then use scripts to copy specific configuration data to remote systems or include these configuration files with the SystemEDGE files when using third-party deployment tools.

The AdvantEDGE View interface enables configuration of individual agents or groups of agents. For more information, refer to the AdvantEDGE View Web Help.

You can also update the SystemEDGE configuration file remotely through SNMP Set commands. This allows programs like AdvantEDGE View, eHealth, HP OpenView, and other SNMP-compliant software to configure the SystemEDGE agent remotely.

Licensing Software Remotely

Automated, remote licensing of the SystemEDGE agent is available through the AdvantEDGE View interface. For more information, refer to Chapter 3, “Licensing the SystemEDGE Agent” and to the AdvantEDGE View Web Help.

Security Issues

System security is a complex problem that involves tradeoffs between usability and system integrity. Making a system more secure often infringes on the usability and the ease of use of the underlying system. Using an SNMP agent such as the SystemEDGE agent requires some policy decisions about what functionality to permit or restrict. This section identifies some of the security issues that you should consider when deploying the SystemEDGE agent.

Those security issues center on four main points:

- Integrity of the underlying system
- SNMPv1 communities
- Scripts and commands invoked by the SystemEDGE agent
- MIB groups

Security of the underlying system is important for restricting access to the SystemEDGE agent configuration files (configuration, monitor, and license) to only those users who are authorized to read and write them. Both read and write access to these files should be restricted:

- Read access can provide information about valid community names and their respective permissions (read-only or read-write).
- Write access to these files allows the modification of community names and associated privileges as well as the modification of self-monitoring table entries, which could be used to execute arbitrary commands.

The SystemEDGE agent currently supports SNMPv1 which uses community-based security. Communities, like a password in some respects, are transmitted in clear-text in SNMP PDUs. Consequently, community names can be vulnerable to packet snooping. Communities have read-only or read-write permissions associated with them which allow the inspection of or inspection/alteration of MIB variables respectively. The SystemEDGE agent can attach IP-address based access-control lists to communities but IP-spoofing can circumvent them. For more information, refer to “Configuring Access Communities” on page 128.

Security Issues with Extension Variables and Action Commands

Because the SystemEDGE agent runs as root or administrator, you must be careful when using extension variables and action commands. More specifically, you should take steps to ensure that all commands and scripts use absolute pathnames, are fully debugged, and contain no ambiguous code or unnecessary options. You can also specify that the agent run all subprograms (for example, actions, extension objects, and so on) as with users and groups other than root. For more information about configuring user and group permissions for sub-programs, refer to “Configuring User/Group Permissions for Subprograms (UNIX Only)” on page 142.

For more information about using extension variables, refer to Chapter 16, “Adding Custom MIB Objects.” For more information about actions that you can use with the SystemEDGE agent, refer to Chapter 10, “Configuring Threshold Monitoring” and Chapter 13, “Configuring Log File Monitoring.”

Security Issues with MIBs

MIB groups within the Host Resources and Systems Management MIBs may permit access to information that is deemed inappropriate by local system policies. Before you deploy the SystemEDGE agent, examine these MIBs to ensure

that they are not violating local security policies. For more information about the Systems Management and Host Resources MIBs, refer to Chapter 7, “Systems Management MIB” and Chapter 9, “Host Resources MIB.”

In particular, verify the settings for access to users, groups, and Who Table information, as well remote-shell support. You can turn off support for these capabilities through the SystemEDGE agent configuration file (`sysedge.cf`). For more information, refer to “Configuring Support for User and Group Information” on page 133 and “Configuring Support for Who Table Information” on page 132.

Troubleshooting and Usage Suggestions

This chapter presents helpful tips for using the SystemEDGE agent. For explanations of the error and warning messages that the SystemEDGE agent and its associated utilities provide, refer to Appendix A.

For the most current information, refer to the eHealth Support Web site at <http://support.concord.com>. You can also send license requests to productlicensing@ca.com.

Using diagsysedge.exe

You can use the diagsysedge.exe program to verify that the agent is running and to obtain information about the agent that you can use for troubleshooting.

NOTE

The examples in this section assume that the agent has a read community of public and is on port 161 or 1691.

Determining Whether the Agent Is Running

You can run diagsysedge.exe with the -b option to determine whether the agent is running.

To determine whether the agent is running:

1. Change to the sysedge/bin directory

2. Enter the following:

```
./diagsysedge.exe -b
```

The command provides output similar to the following:

```
#./diagsysedge.exe -b
Copyright (C) 2003 Concord Communications, Inc. All rights reserved.
http://www.concord.com http://www.concord.com/sysedge

diagsysedge: (V1.01.0 - LINUX)

egyptian
2.2.14-5.0, Red Hat Linux release 6.2 (Zoot)#1 Tue Mar 7 21:07:39 EST 2000
Concord Communications, Inc. SystemEDGE Agent Version 4.1 Patchlevel 4
```

If the agent is *not* running, you receive an error message that the SNMP operation failed. Attempt to start the agent again, using the instructions in Chapter 5, “Starting the SystemEDGE Agent,” for your operating system.

Obtaining a Report for Troubleshooting

If you are encountering problems with your SystemEDGE agent, Technical Support may ask you to run `diagsysedge.exe` to generate a report that they can use for troubleshooting.

To generate a report that you can send to Technical Support:

1. Change to the `sysedge/bin` directory
2. Enter the following:

```
./diagsysedge.exe
```

The command creates a `diagsysedge.txt` file that provides information similar to the following:

- Version of the `diagsysedge.exe` program
- Current date and time

- IP address of the system on which you are running the agent
- Operating system platform
- Community string
- Port on which the agent is running
- Timeout value
- Location of the sysedgeddiag.txt output file
- Contents of the sysedge.cf, sysedge.lic, and sysedge.mon files (from the /etc [UNIX] or %SystemRoot%\System32 directory [Windows])
- Information about registry settings that the setup program modified
- A detailed System Information report

If the agent is *not* running, you receive an error message that the SNMP operation failed. Attempt to start the agent again, using the instructions in this chapter for your operating system.

Common Problems and Questions

This section describes problems that might occur when you are installing and using the SystemEDGE agent. It also provides instructions for resolving these problems. This section is organized by observable symptoms of a problem, and questions for each symptom to help you isolate the cause.

Agent Not Responding to SNMP Requests

When the agent is not responding to SNMP requests, the NMS software cannot query the SystemEDGE agent. To resolve this problem, you must verify that the agent is running, that it is properly configured, and that the management system software is properly configured.

Is the SystemEDGE Agent Running?

You can verify that the agent is running in one of the following ways: using the `diagsysedge.exe` program, or using `netstat`. For information about using `diagsysedge.exe`, refer to “Determining Whether the Agent Is Running” on page 423.

To verify that the agent is running with `netstat`:

1. Enter `netstat -a` and look for UDP/161, UDP/1691, or SNMP.
2. Enter `ps -aux` or `ps -aef` and look for `sysedge`.
3. Examine system log files for agent error messages.
 - On Windows, examine `%SystemRoot%\system32\sysedge.log`.
 - On UNIX, examine the `syslog` files.

Is the SystemEDGE Agent Starting at System Initialization?

To verify that the agent is started at system initialization:

- Check the UNIX startup script.
- Make sure that the SystemEDGE service is configured for automatic startup.

Is the SystemEDGE Agent Responding to Queries?

Use the `sysvariable` utility to query a system and prove that the agent is responding to queries. You must specify the port number (unless you are using the default port of 161) for the SystemEDGE agent and the community string for your system.

For example, to display the operating system release number for a UNIX system where the SystemEDGE agent is on port 1691 and the community string is `public`, enter the following:

```
#./sysvariable 127.0.0.1:1691 public -r
```

For a Windows system where the SystemEDGE agent is on port 1691 and the community string is public, enter the following:

```
C:\sysedge\bin>sysvariable.exe 127.0.0.1:1691 public -r
```

Is the SystemEDGE Agent Licensed Correctly?

To verify that the agent is licensed correctly, refer to “SystemEDGE Agent Not Accepting Valid License” on page 429.

Are the SystemEDGE Agent Binaries and Configuration Files Installed?

To verify that the correct agent binaries and configuration files are installed, refer to “Agent Does Not Run on A Particular Operating System Version” on page 430.

Is the SystemEDGE Agent Co-Existing with Other Agents?

If you are using other agents with the SystemEDGE agent, verify their coexistence.

```
setup.exe -c -v
```

Is the SystemEDGE Agent Configured Correctly?

To verify that the agent is configured correctly, check the communities and access control lists:

- On UNIX systems, examine /etc/sysedge.cf for community strings and access control lists. If an alternative configuration file is in use, verify that the agent is using that alternative configuration file.
- On Windows systems, verify access controls and communities through the Network Control Panel. For more information, refer to Chapter 4, “Configuring the SystemEDGE Agent.”

Is the Management System Software Configured Correctly?

To verify that the management system software is correctly configured, check to see that it is querying the correct system and port number if the SystemEDGE agent is running on an alternate UDP port (for example, on UDP/1691).

Management System Not Receiving SNMP Trap Messages

If the management system is not receiving SNMP traps from the SystemEDGE agent, you must verify that the agent is sending Trap PDUs and that it is sending them to the correct addresses. If you have already verified those conditions, the problem is most likely due to misconfiguration of the management system.

Is the SystemEDGE Agent Running?

To verify that the agent is running, refer to “Agent Not Responding to SNMP Requests” on page 425.

Is the SystemEDGE Agent Correctly Configured to Send Trap Messages?

To verify that the agent is configured to send trap messages to the appropriate addresses, do one of the following:

- On UNIX systems, verify the trap communities in the `sysedge.cf` file. For example, make sure that only one IP address is specified for each trap community statement. For more information on trap community configuration, refer to Chapter 4, “Configuring the SystemEDGE Agent.”
- On Windows systems, verify the trap communities through the Control Panel Network application. For more information, refer to “Configuring Trap Communities” on page 130.

Is the SystemEDGE Agent Sending Traps?

To verify whether the SystemEDGE agent is sending traps, do the following:

- On UNIX systems, examine the syslog log files.
- On Windows systems, examine
%SystemRoot%\system32\sysedge.log.

Is the Management System Configured Correctly?

To verify that the management system has been properly configured, make sure that the empire.asn1 file has been imported into the management station so that it knows the format of SNMP Trap PDUs that are generated by the SystemEDGE agent. For more information about troubleshooting the management system software, contact your management system vendor.

SystemEDGE Agent Not Accepting Valid License

If the SystemEDGE agent is not operating, even if a valid license exists in the agent's license file, you must verify that the agent is reading the correct license file, that the license was correctly entered, and that you supplied the correct license information supplied to CA.

Is the SystemEDGE Agent Reading the Correct License File?

To verify that the agent is reading the correct license file, do the following:

- On UNIX systems, the correct file is /etc/sysedge.lic by default.
- On Windows systems, the correct file is
%SystemRoot%\system32\sysedge.lic.

Was the License Entered Correctly into the License File?

To verify that the license is correct, compare the license key in the license file to the one that was provided by CA. For example, verify that O's and 0's, and l's and 1's were not transposed.

Was the Correct License Information Supplied to CA?

Verify that the license information you supplied to CA matches your system. This information (which was generated by the SystemEDGE agent) takes the following form:

```
SystemEDGE jupiter NTx86 4.0 346561363366b19c 4.1 Patchlevel 1
```

For more information about licensing, refer to Chapter 3, “Licensing the SystemEDGE Agent.”

Agent Does Not Run on A Particular Operating System Version

If the System agent was installed on an operating system on which it does not run, the agent displays a message similar to the following when it starts for the first time after installation:

```
This agent binary is compiled for X, not Y
```

Because the SystemEDGE agent works closely with the underlying operating system, each release may require different binaries. For example, kernel data structures and application programmer interfaces may change from release to release, sometimes in subtle ways.

Therefore, for versions of Solaris 2.x and some versions of HP-UX 10.x, the SystemEDGE agent uses different binaries for each release. The error message displayed above indicates that you are attempting to execute a SystemEDGE agent binary on an operating system release for which it is not compiled.

To ensure that your system is executing the correct SystemEDGE agent binary, examine the UNIX startup script to ensure that it is selecting the appropriate binary. Table 83 lists the appropriate binaries for the software versions that this Release of the SystemEDGE agent supports.

Table 83. SystemEDGE Agent Binaries (Page 1 of 2)

Operating System Version	SystemEDGE Agent Binary
AIX 4.3.3	sysedge.aix433
AIX 5.1 and 5.2 (32-bit)	sysedge.aix51
AIX 5.1 and 5.2 (64-bit)	sysedge.aix51.64
Digital UNIX 4.0F or 4.0G, or	sysedge.du4
Tru64 5.1A	sysedge.du5
HP-UX 11.0 and 11i (32-bit)	sysedge.hpux11
HP-UX 11.0, 11i, and 11i v2 (64 bit)	sysedge.hpux11-64
Linux	sysedge
Solaris 2.6	sysedge.sol26
Solaris 2.7 (32-bit)	sysedge.sol27
Solaris 2.7 (64-bit)	sysedge.sol27-sparcv9
Solaris 8 (32-bit)	sysedge.sol28
Solaris 8 (64-bit)	sysedge.sol8-sparcv8
Solaris 9 (32-bit)	sysedge.sol9
Solaris 9 (64-bit)	sysedge.sol9-sparcv9
Solaris 10 (64-bit)	sysedge.sol210-sparcv9
Solaris 10 (AMD64)	sysedge.sol210-x86-64
Solaris x86 2.6 (32-bit)	sysedge.sol26

Table 83. SystemEDGE Agent Binaries (Page 2 of 2)

Operating System Version	SystemEDGE Agent Binary
Solaris x86 2.7 (32-bit)	sysedge.sol27
Solaris x86 2.8 (32-bit)	sysedge.sol28
Solaris x86 2.9 (32-bit)	sysedge.sol29
Solaris x86 2.10 (32-bit)	sysedge.sol210
Windows NT 4.0, Windows 2000, Windows XP, and Windows 2003	sysedge.dll

Bind Failed: Address Already In Use

If a bind fails because the address is in use, the SystemEDGE agent displays a message similar to the following when it first starts:

```
sysedge: bind call failed: Address already in use
sysedge: another agent is probably running on port X
```

This message most often occurs when another SNMP agent is already up and running and is bound to port UDP/161, which is the default, well-known SNMP agent port.

For more information about using the SystemEDGE agent with other SNMP agents, refer to Chapter 6, “Using the SystemEDGE Agent with Other SNMP Agents.” For more information about using SystemEDGE on an alternative UDP port, refer to Chapter 5, “Starting the SystemEDGE Agent.”

Updating the Monitor Configuration File

The `sysedge.mon` monitor configuration file serves as non-volatile backing store to the agent's in-memory self-monitoring tables. When SNMP updates are made to any of the agent's in-memory self-monitoring tables, those tables are written to `sysedge.mon`. Consequently, the agent does not interpret updates to `sysedge.mon` that occur while it is running until it is restarted. Also, updates made to `sysedge.mon` while the agent is running may be lost if the agent writes its in-memory monitor table over the tables in `sysedge.mon`. In addition, configuration file directives in `sysedge.cf` take precedence over those in `sysedge.mon`.

To update the SystemEDGE agent's monitor configuration file:

1. Open a command prompt.
2. Stop the agent.
 - On Windows, enter the following:

```
net stop sysedge
```

- On Solaris, enter the following when you are logged in as root:

```
/etc/rc2.d/S99sysedge stop
```

NOTE

For other UNIX operating systems, enter the path and name for your SystemEDGE startup script. For a list of startup script names, refer to “SystemEDGE Directories and Files for UNIX Systems” on page 90.

3. Use a text editor to update sysedge.mon.
4. Restart the agent.
 - On Windows, enter the following:

```
net start sysedge
```

- On Solaris, enter the following when you are logged in as root:

```
/etc/rc2.d/S99sysedge start
```

NOTE

For other UNIX operating systems, enter the path and name for your SystemEDGE startup script. For a list of startup script names, refer to “SystemEDGE Directories and Files for UNIX Systems” on page 90.

5. Verify that the updated `sysedge.mon` file does not contain any errors. To do so, examine the `sysedge.log` file on Windows, or the `syslog` file output on UNIX for error messages pertaining to the monitor configuration file.

NOTE

You can update the `sysedge.mon` file directly, but you should not need to: configuration file directives exist to create all supported types of `sysedge.mon` entries.

If Entries Are Not Being Added to `sysedge.mon`

If you are adding entries to `sysedge.mon` through SNMP and find that the file is not being updated, it is possible that the file was at some point changed to have read-only permissions.

When `sysedge.mon` is read-only and you attempt to add new entries, the following occurs:

- `sysedge.mon` regains read-write permissions.
- Two new files are created:
 - `sysedge.new` contains the current content of `sysedge.mon` plus any new entries you added from the time it was set to read-only.
 - `sysedge.old` contains the read-only version of `sysedge.mon`.

To resolve the problem and commit your changes to `sysedge.mon`:

1. Verify that the `sysedge.mon`, `sysedge.new`, and `sysedge.old` files exist in the `systemRoot\system32` directory.
2. Enter the following commands at the command line:

```
Attrib -r sysedge.*
Del systemRoot\system32\sysedge.old
Del systemRoot\system32\sysedge.mon
Move systemRoot\system32\sysedge.new sysedge.mon
```

Automatically Restarting Processes

The SystemEDGE agent can perform actions when an entry in any of the agent's self-monitoring tables evaluate to True. You can use this capability to restart processes, clean up file systems, and so on.

To set the SystemEDGE agent to restart processes when they fail:

1. Create an entry in the Monitor table to monitor a particular application or process. (You can create an entry through monprocess, through an SNMP management station, or by editing the sysedge.mon file in a text editor.)
2. Make sure that entry contains an action script. That action script is responsible for restarting the application and resetting the monitor-table entry for the application or process. The SystemEDGE agent distributions come with an example script, restartproc.sh, that performs this function.

When the process fails, the monitor table entry evaluates to True, sends an SNMP trap to the management system, and executes the action script. The action script restarts the application, waits a few seconds, and then calls monprocess to reinitialize the monitor-table entry so that the agent will resume monitoring the new process.

Implementing Trap Severity Levels

Currently, trap messages generated by the SystemEDGE agent contain many variables, but not an explicit severity level. However, trap messages that are generated by the self-monitoring tables contain an ASCII description field. You can implement severity levels if they are encoded within the corresponding description string for the particular table row in question.

Using this approach, you can assign severity levels like Critical or Warning, and you can include those severity levels within the table-description field. For example, you can use the description in conjunction with file-system monitoring to indicate that a Critical event occurs when the file system is over 95% full with a message similar to the following: “Critical: / filesystem over 95% full.”

When management software receives a trap message, it can easily identify the severity level using the description string that is already included within applicable trap messages. The description string provides an easy-to-read method for indicating severity rather than one that requires specialized management-station coding or modification.

Required and Recommended System Patches

The SystemEDGE agent requires few, if any, system patches for proper functioning. Unless your platform and operating system release are mentioned in this section, you do not need any known patches. For the latest information about patches, refer to the eHealth Support Web site (<http://support.concord.com>).

The following patch is recommended for Solaris 2.6 systems:

Solaris 2.6 – 105742 updates the le drivers to better support MIB-2. Requires patch 105181-05 or newer. Patch 105795 fixes the hme driver so that it indicates ifSpeed.

The following patches are recommended for HP-UX systems:

- HP-UX 11.0 (64-bit) – XSWGR1100 revision B.11.00.49.3 or later is required, or patch PHNE 19984. This bundle provides a missing system library.
- HP-UX 11.11 – PHNE 27063 s700 800 11.11 cumulative ARPA Transport patch or later is recommended. This patch fixes memory issues with HP-UX libraries.

Error Messages

This chapter describes error messages you may encounter while using the SystemEDGE agent. The error messages are organized by type into the following groups:

- SystemEDGE agent error messages
- Command-line utility error messages

SystemEDGE Agent Error Messages

This section describes the error messages that you may encounter when using the SystemEDGE agent. Depending on the error, the message will either be printed to standard error or logged through the syslog utility. The messages in this appendix are sorted alphabetically.

action execution failed

The SystemEDGE agent failed to execute an action command due to an error. Check your action script name and arguments.

authenFailure src: A.B.C.D community X

The SystemEDGE agent has sent an SNMP authentication failure trap message because it does not permit SNMP queries using community *x* from IP address *A.B.C.D*.

bad ioconfig magic

Upon startup, the SystemEDGE agent uses whatever operating system support is available to attempt to discover the devices that are contained in the underlying system. This error message indicates the SystemEDGE agent could not properly read the HP-UX file `etc/ioconfig` and will, therefore, attempt to discover devices without the operating system's aid.

bad pid X for write_runStatus

An attempt to change the Host Resources `hrRunStatus` variable for a process has failed due to an invalid process ID (PID). Double-check the PID value for that process and try management operation again.

bad shellOutput directory X

The directory that is specified for `remoteShell` invocation output is invalid. All output from `remoteShell` invocations should be placed in `/tmp` or `\tmp`.

bad shellOutput file X

The designated `remoteShell` output file is invalid because it is either not a file (for example, it is a directory) or it is contained in an invalid directory (for example, not in `/tmp`).

bind failed

The SystemEDGE agent attempted to bind to a UDP port and failed. If that UDP port is already in use, another SNMP agent is already running on that system and is using that port. This problem commonly occurs when multiple agents attempt to use the same UDP port (for example, UDP/161). The Internet Assigned Numbers Authority (IANA) has reserved UDP/1691 for the SystemEDGE agent. You can configure the SystemEDGE agent to use that port by specifying it on the command line through the `-p` option.

block size for X is 0, using 1K

The SystemEDGE agent failed to determine a file system's underlying block size and is using 1000 bytes as a default block size for calculations in the Systems Management MIB Mounted Devices table.

cant open kmem

The SystemEDGE agent could not open the UNIX device `/dev/kmem`, which allows the agent to read kernel memory. Verify that `/dev/kmem` exists and that the agent has permission to read it. Running the SystemEDGE agent as a user other than the root user could cause this problem.

cannot find title index for

On Windows, the SystemEDGE agent could not properly initialize the relevant information to read a particular performance variable from the Windows registry. Only this variable is affected. All other Windows variables should be supported.

cannot locate disk pstat data, diskStatsTable not supported

On HP-UX systems, the SystemEDGE agent was unable to locate the appropriate disk statistics through an HP-UX application programmer interface (API). Disk statistics for the particular drive are, therefore, not supported. Disk statistics for other drives, however, should be unaffected.

cant open kmem

On UNIX systems, the SystemEDGE agent reads the device `kmem` to retrieve certain kernel parameters and statistics. This error message indicates that the agent could not open the `/dev/kmem` device perhaps due to permissions problems. Verify that the agent has read access to `/dev/kmem`.

cant open socket for mib-2

On some UNIX systems, the SystemEDGE agent uses a socket to obtain MIB-II statistics. This error message indicates that the agent was unable to obtain such a socket. Most likely, the agent will be unable to support MIB-II related statistics but should otherwise operate normally.

caught SIGHUP

The SystemEDGE agent caught a UNIX hang-up signal and is continuing to operate normally.

config file error in subprogram_user_name directive

The SystemEDGE agent configuration file contains an error in the subprogram directive. The directive will be ignored. Fixing the statement and restarting the agent will correct the problem.

config syntax error, line X

The SystemEDGE agent has encountered a syntax error in its configuration file at line *x*. The offending line will be ignored and the rest of the configuration file will be parsed.

Could not find a valid license for machine X

The SystemEDGE agent could not find a valid license for the system on which it is operating. Check that the SystemEDGE agent license file is in the correct location and contains a valid license. On Windows systems, the configuration file is located in the system root directory, %SystemRoot%\system32\. On UNIX systems, the SystemEDGE agent looks for the file in the /etc directory.

couldn't fork sub-shell

The SystemEDGE agent could not create or invoke a sub-shell to process a remoteShell operation. If the system process table is full, additional processes cannot be created.

counterType X not supported. Entry will not be added.

The SystemEDGE agent could not support registry extension objects of type *x* because the agent cannot understand it. (There is no way to map it to an SNMPv1 object type.) The agent will not support this registry extension object.

CreateEvent for traps failed

On Windows systems, the SystemEDGE agent must create an internal operating system event resource for use with sending SNMP traps. This error message indicates that the event creation failed. Consequently, the SystemEDGE agent may not be able to send private-enterprise trap messages.

createMutex failed for query mutex

On Windows systems, the SystemEDGE agent uses an operating system mutex resource for controlling access to structures shared with the master agent. Consequently, the SystemEDGE agent may not be able to operate properly.

createMutex failed for result mutex

On Windows systems, the SystemEDGE agent uses an operating system mutex resource for controlling access to structures shared with the master agent. Consequently, the SystemEDGE agent may not be able to operate properly.

/dev/lan is missing. SystemEDGE cannot continue.

The SystemEDGE agent requires a /dev/lan device to run on HP-UX systems. It will not start if it cannot find that device. Instead, it will log a message to syslog and exit. For information about the device, refer to your HP-UX documentation.

dkiotime read failed, no disk stats

The SystemEDGE agent was unable to read a disk statistic structure of the kernel. Consequently, disk statistics may not be supported.

discovering HP-UX devices by hand

On HP-UX systems, the SystemEDGE agent is discovering devices manually rather than through automated techniques. This discovery method is used on older versions of HP-UX, which do not support those automated methods.

empire_agent_init failed

On Windows systems, the SystemEDGE agent failed to initialize due to licensing errors, invalid or non-existent configuration files, or some other problem. The SystemEDGE agent will normally print additional, more specific error messages that indicate the cause of the problem.

error parsing X in monitor file, line Y

The SystemEDGE agent encountered an error while parsing the sysedge.mon monitor configuration file. It prints the offending line number.

event regcomp failed, X *desc_filt*

The SystemEDGE agent failed to compile a Windows event-monitoring regular expression because it was invalid. Verify the regular expression and try the management operation again.

executing subprograms as group X

On UNIX systems, the SystemEDGE agent is executing all subprograms with permissions of group *x*, as specified in a configuration file directive.

executing subprograms as user Y

On UNIX systems, the SystemEDGE agent is executing all subprograms with permissions of user *y*, as specified in a configuration file directive.

execv failed for action

The SystemEDGE agent failed to exec the action command. Exec failures can occur if the command is invalid, if the binary or shell script no longer exists in the specified location, or if the execute permissions are not properly set. Check the action command (for example, by hand) and, if appropriate, update the SystemEDGE agent configuration file, *sysedge.cf*.

execv failed for extension command

The SystemEDGE agent failed to exec the extension command. Exec failures can occur if the command is invalid, if the binary or shell script no longer exists in the specified location, or if the execute permissions are not properly set. Check the extension command (for example, by hand) and if appropriate, update the SystemEDGE agent configuration file.

extension command file X is not a regular file

The extension command file is not a valid executable or shell script. Check the specified file and ensure that it is not a directory, device, and so on. For more information about extending the SystemEDGE agent, refer to Chapter 16, “Adding Custom MIB Objects.”

extension filename too long

The extension command file name is too long. The SystemEDGE agent limits command file names to 256 characters. Check the extension statement in the configuration file and restart the SystemEDGE agent. For more information about extending the SystemEDGE agent, refer to Chapter 16, “Adding Custom MIB Objects.”

extension variable X already in use

The extension command variable specified in the configuration file is already in use. Extension commands must specify a number that has already been used. Check the configuration file duplicate extension command numbers and restart the SystemEDGE agent. For more information about extending the SystemEDGE agent, refer to Chapter 16, “Adding Custom MIB Objects.”

failed to add monitor entry index X

The SystemEDGE agent failed to add a Monitor table entry. This error usually results from bad parameters, which are most often due to invalid intervals, boolean operators, object identifiers, and so on. Fix the monitor entry and restart the SystemEDGE agent. For the correct syntax for Monitor table entry commands, refer to Chapter 10, “Configuring Threshold Monitoring.”

failed to alloc anIDE struct

The SystemEDGE agent failed to allocate memory for an IDE device, most likely because the underlying system is low on memory. The SystemEDGE agent will continue to operate.

failed to alloc space for monitor

The SystemEDGE agent failed to allocate memory for its Monitor table, most likely because the underlying system is low on memory. The SystemEDGE agent will continue to operate, but it will not be able to perform self-monitoring.

failed to create a trap session

The SystemEDGE agent failed to allocate the resources necessary to send SNMP trap messages, most likely because of an error in the underlying system. The SystemEDGE agent will continue to operate, but will not be able to send SNMP trap messages.

failed to create timer event, X

The SystemEDGE agent on a Windows system failed to create a Windows timer event necessary for sub-agent initialization and operation. The problem is most likely due to an error on the underlying Windows system. Unfortunately, problems such as this often require you to reboot the Windows system.

failed to create a trap session

The SystemEDGE agent failed to create an internal session to use for sending SNMP traps. Consequently, the agent may not be able to send SNMP traps.

failed to create timer event

On Windows systems, the SystemEDGE agent uses a timer construct to perform periodic processing. This message indicates that the agent failed to allocate such a resource. Consequently, the agent's self-monitoring capabilities may not function properly.

failed to create trap pdu

The SystemEDGE agent failed to create an SNMP trap message for transmission to SNMP management systems. This error usually occurs when the underlying system is low on memory. The SystemEDGE agent will not send the particular SNMP trap message, but it will continue to attempt to send subsequent SNMP trap messages.

failed to get dkscinfo

The SystemEDGE agent was unable to get a disk statistics structure out of the UNIX kernel. Consequently, disk statistics may not be supported for some or all disks.

failed to get domain name

The SystemEDGE agent was unable to determine the system's DNS domain name. This error usually occurs when the underlying system's DNS domain name is not configured. This error has little effect on the SystemEDGE agent's overall operation.

failed to get service handle

The SystemEDGE agent was unable to obtain a Windows service handle. On Windows, service information is obtained through an operating-system specific resource termed a handle. This message indicates that there was a problem obtaining a particular service's handle. The SystemEDGE agent will continue operation but may not be able to provide all information about a particular Windows service.

failed to open /dev/netman

On HP-UX systems, the SystemEDGE agent was unable to open the device `/dev/netman`, which allows the agent to support MIB-II. Verify that the `/dev/netman` file exists and that the SystemEDGE agent has permission to read it. Running the SystemEDGE agent as a user other than the root user can cause this problem.

failed to open /system

On HP-UX 9.x systems, the SystemEDGE agent failed to open the `/system` directory for inspection of locally installed software packages and patches. Check that the `/system` directory exists and that the SystemEDGE agent has permission to read it. This

problem can occur if you are running the SystemEDGE agent as a user other than the root user. If you do not correct this error, the SystemEDGE agent will continue to operate but will not be able to support the Host Resources Installed Software table.

failed to open /var/adm/sw/products

On HP-UX systems, the SystemEDGE agent failed to open the /var/adm/sw/products directory for inspection of locally installed software packages. Verify that the /var/adm/sw/products directory exists and that the SystemEDGE agent has permission to read it. This problem can occur if you are running the SystemEDGE agent as a user other than the root user. If you do not correct this error, the SystemEDGE agent will continue to operate but will not be able to support the Host Resources Installed Software table.

failed to open /var/sadm/patch

On Solaris 2.x systems, the SystemEDGE agent failed to open the /var/sadm/patch directory for inspection of locally installed patches. Check that the /var/sadm/patch directory exists and that the SystemEDGE agent has permission to read it. This problem can occur if you are running the SystemEDGE agent as a user other than the root user. If you do not correct this error, the SystemEDGE agent will continue to operate but will not be able to determine which patches have been installed on the underlying system.

failed to open /var/sadm/pkg

On Solaris 2.x systems, the SystemEDGE agent failed to open the /var/sadm/pkg directory for inspection of locally installed software packages. Check that the /var/sadm/pkg directory exists and that the SystemEDGE agent has permission to read it. This problem can occur if you are running the SystemEDGE agent as a user other than the root user. If you do not correct this error, the SystemEDGE agent will continue to operate but will not be able to support the Host Resources Installed Software table.

failed to open config file X

The SystemEDGE agent failed to open the specified configuration file (*x*). Verify that the *x* file exists and that it is readable by the SystemEDGE agent. The SystemEDGE agent will not operate until you fix this problem.

failed to open ioconfig

Upon startup, the SystemEDGE agent attempts to discover the devices contained in the underlying system using any operating system support that is available. This error message indicates that the SystemEDGE agent could not properly read the HP-UX file `/etc/ioconfig` and will, therefore, attempt to discover devices without the operating system's help.

failed to open ip for mib2

On Solaris 2.x systems, the SystemEDGE agent needs access to the `/dev/ip` file in order to properly support MIB-II. Check that the `/dev/ip` file exists and that it is readable by the SystemEDGE agent. This problem can occur if you are running the SystemEDGE agent as a user other than the root user.

failed to open kmem

On UNIX systems, the SystemEDGE agent reads the `kmem` file to retrieve certain kernel parameters and statistics. This error message indicates that the agent could not open the `/dev/kmem` file, perhaps because of permissions problems. Verify that the agent has read access to `/dev/kmem`.

failed to open mnttab file

On UNIX systems, the SystemEDGE agent was unable to open the system file that indicates which file systems were mounted and are accessible to users. Although the SystemEDGE agent will continue to operate, it may be unable to answer SNMP queries regarding file systems, disks, and partitions. Verify that the system-specific mounted file system file exists and that it is readable by the SystemEDGE agent.

failed to open/create mon file

The SystemEDGE agent was unable to open or create a monitor configuration file. When the SystemEDGE agent's in-memory monitor table changes, it is written out, in ASCII format, to the monitor table configuration file. This message indicates that the operation failed. The SystemEDGE agent will continue to operate and will continue to monitor MIB objects based on its in-memory monitor table. However, that contents of the in-memory monitor table may be lost if the agent is restarted before it can properly save the data.

failed to open openprom device

On Sun systems, the SystemEDGE agent obtains some configuration information from the openprom facility. This error message indicates that the agent was unable to open that file; consequently, some configuration information may not be supported.

failed to parse config file

The SystemEDGE agent was unable to parse the configuration file. Verify that the configuration file exists, is in either the default location or the location that you specified on the command line, and is readable. The SystemEDGE agent will not operate if it cannot find a valid configuration file.

failed to push ARP for mib2

On Solaris 2.x, the SystemEDGE agent needs access to the `/dev/ip` file and the arp Streams module in order to properly support MIB-II. Verify that the `/dev/ip` file exists and that it is readable by the SystemEDGE agent. This problem can occur if you are running the SystemEDGE agent as a user other than the root user.

failed to push TCP for mib2

On Solaris 2.x systems, the SystemEDGE agent needs access to the `/dev/ip` file and the `tcp` Streams module in order to properly support MIB-II. Verify that the `/dev/ip` file exists and that it is readable by the SystemEDGE agent. This problem can occur if you are running the SystemEDGE agent as a user other than the root user.

failed to push UDP for mib2

On Solaris 2.x, the SystemEDGE agent needs access to the `/dev/ip` file and the `udp` Streams module in order to properly support MIB-II. Verify that the `/dev/ip` file exists and that it is readable by the SystemEDGE agent. This problem can occur if you are running the SystemEDGE agent as a user other than the root user.

failed to read ioconfig magic

Upon startup, the SystemEDGE agent attempts to discover the devices that are contained in the underlying system using whatever operating system support is available. This error message indicates the SystemEDGE agent could not properly read the HP-UX `/etc/ioconfig` file and will, therefore, attempt to discover devices without the operating system's help.

failed to read monitor file

The SystemEDGE agent failed to open either the default monitor file or the file that you specified on the UNIX command line. The default monitor file for UNIX is `/etc/sysedge.mon`; for Windows, it is `%SystemRoot%\system32\sysedge.mon`.

failed to reload utmp cache

The SystemEDGE agent failed to reload its internal table of users who are currently logged in to the system. The SystemEDGE agent will continue function normally, but may be unable to answer SNMP queries of Who Table objects.

failed to rename mon file

The SystemEDGE agent periodically writes its in-memory monitor table to the sysedge.mon file. Before doing so, the SystemEDGE agent renames the current sysedge.mon to sysedge.old (in the same directory as the current file). This error message indicates that the rename operation failed. The SystemEDGE agent will continue to function normally, but the contents of the old monitor table will not be recoverable.

failed to allocate history entry

The SystemEDGE agent failed to allocate memory for a history table entry. It will therefore not create a history table entry, and the configuration file statement will be ignored. The SystemEDGE agent will, however, continue to parse the remainder of the configuration file.

failed to send COLDSTART trap

The SystemEDGE agent failed to send a MIB-II defined Cold Start trap message to its SNMP management systems. This error usually results from underlying operating system problems. The SystemEDGE agent will continue to operate normally and will attempt to continue to send SNMP trap messages as necessary.

fork failed for extension command

The SystemEDGE agent failed to fork itself in order to execute the extension command. Fork failures occur if the system has insufficient resources to create new processes. This error indicates that no extension command was executed.

fork failed for monitor action

The SystemEDGE agent failed to fork itself in order to execute the monitor action. Fork failures occur if the system has insufficient resources to create new processes. This error indicates that no monitor action was executed, but the monitor table row is still active and will continue to attempt action commands when the table row evaluates to True.

fork failed for logmonitor action

The SystemEDGE agent failed to fork itself in order to execute the Log Monitor action. Fork failures occur if the system has insufficient resources to create new processes. This error indicates that no Log Monitor action was executed, but the Log Monitor table row is still active and will continue to attempt action commands when the table row evaluates to True.

FPE signal caught

The SystemEDGE agent caught a floating-point exception error, probably because of a divide-by-zero error. Report this message to eHealth Technical Support.

ID is X,Y

On Windows systems, the SystemEDGE agent reports its process (X) and thread (Y) identifiers in its log file. This message is for debugging purposes only and can be ignored.

identical threads IDs

On Windows systems, the SystemEDGE agent has discovered two separate, distinct threads with the same thread identifier. Although this condition is technically impossible, it can still occur. This message indicates that the SystemEDGE agent has discovered this situation and has properly accommodated it.

invalid extension variable access mode

The extension statement in the SystemEDGE agent configuration file contains an invalid access mode. Valid access modes are read-only or read-write, indicating whether an extension variable can be only read or read or written. The agent will ignore the offending extension statement and will parse the remainder of the configuration file. For more information about extending the SystemEDGE agent, refer to Chapter 16, “Adding Custom MIB Objects.”

invalid extension variable type X

The extension statement in the SystemEDGE agent configuration file contained an invalid SNMP type. Valid extension variable SNMP types include the following: integer, counter, gauge, octetstring, timeticks, objectid, and ipaddress. The agent will ignore the offending extension statement and will parse the remainder of the configuration file. For more information about extending the SystemEDGE agent, refer to Chapter 16, “Adding Custom MIB Objects.”

invalid history description

An emphistory configuration file statement contained an invalid description field. Verify that the description is delineated by single quotation marks. The agent will not create a history table entry and will ignore the statement. The agent, will, however, continue to parse the remainder of the configuration file.

invalid history object type

An emphistory configuration file statement contained an object identifier whose base SNMP type is not an integer, counter, or gauge. The SystemEDGE agent will not create a history table entry and will ignore the statement. The agent will, however, continue to parse the remainder of the configuration file.

Invalid monitor table index

An invalid Monitor table index was specified either in a configuration file command or through SNMP row creation to the monitor table. Valid monitor table indexes must be greater than 10. Rows 1 through 10 are reserved by the SystemEDGE agent for internal use.

invalid monprocess regular expression

A monprocess configuration file statement contained an invalid regular expression. The agent will not create the Monitor table entry and will ignore the statement. The SystemEDGE agent, however, will continue to parse the remainder of the configuration file.

invalid NT event log name

The Windows event-log name that was supplied through the configuration file or through the command-line utility was incorrect.

invalid NT event type

The Windows event type that was supplied through the configuration file or through the command-line utility was incorrect.

invalid number history buckets

An emphistory configuration file statement contained an invalid number of history buckets. The SystemEDGE agent will not create a history table entry and will ignore the statement. The agent will, however, continue to parse the remainder of the configuration file.

invalid SNMP variable type X

The SNMP variable type *x*, as specified in the agent's configuration file for registry extension objects, was incorrect. For a list of supported SNMP types, refer to Chapter 16, “Adding Custom MIB Objects.”

lock of mnttab lock failed

The SystemEDGE agent failed to lock the UNIX mounted-device file. Consequently, information about mounted file systems may not be supported.

license file /etc/sysedge.lic not found

The SystemEDGE agent could not find the default UNIX license file, /etc/sysedge.lic. Without a proper license, the SystemEDGE agent will not continue to operate.

license file not found

The SystemEDGE agent could not find a license file that was provided on the command line. Without a proper license, the SystemEDGE agent will not continue to operate.

log file is not regular

The SystemEDGE agent was instructed, either through configuration file statements or remotely through SNMP, to monitor a log file that is not a regular ASCII file. Consequently, the SystemEDGE agent will not monitor the log file for the corresponding regular expression.

log filename too long

A log-file name that was specified in the SystemEDGE agent configuration file exceeds the maximum file name length of 256 characters. Consequently, the SystemEDGE agent will not monitor the log file for the corresponding regular expression.

logmon entry X re-initialized

The Log Monitor entry *x* was automatically reinitialized by the SystemEDGE agent.

logmon regcomp failed, X

A invalid regular expression was configured for log file monitoring, either remotely through SNMP or through logmon configuration file statements. The agent will not add this Log Monitor table entry, but it will continue to monitor other log files for their corresponding regular expressions.

logmon trap entry not ready Index:X

The SystemEDGE agent has sent a Log Monitor trap message that indicates that entry *x* is notReady. The agent will no longer perform log monitoring on behalf of Log Monitor table entry *x*, but will continue to perform log monitoring on all other entries. A Log Monitor table entry can become notReady when an error occurs while reading its log file.

logmon trap Index:X

The SystemEDGE agent has sent a Log Monitor trap message that indicates that it has detected a log file match for entry *x*. The agent will continue to monitor this log file and will send additional trap messages when it finds new matches.

logmonitor action execution failed

The SystemEDGE agent failed to execute a Log Monitor action command. The agent will continue to monitor log files and will attempt to perform action commands when it finds matches.

malloc of trap contents failed

The SystemEDGE agent failed to send a trap message because it was unable to acquire the necessary memory. This error is most likely caused by a lack of memory on the underlying system. The SystemEDGE agent will continue to attempt to send trap messages.

monitor action execution failed

The SystemEDGE agent failed to execute a Monitor table action command. The agent will continue to monitor entries and will attempt to perform action commands when monitor table expressions evaluate to True.

monitor entry X not ready

The SystemEDGE agent has sent a Monitor table trap message that indicates that entry *x* is notReady. The agent will no longer evaluate Monitor table entry *x*, but will continue to perform monitoring on all other entries. A Monitor table entry can become notReady when an error occurs while accessing a MIB variable.

monitor trap Index:X

The SystemEDGE agent has sent a Monitor trap message that indicates that Monitor table row *x* has evaluated to True. If the entry contained an action command, the agent has executed it. The agent will continue to monitor this entry and will send additional trap messages when the entry reevaluates to True.

monprocess requires regular expression

A monprocess statement in the SystemEDGE agent configuration file did not contain a regular expression. monprocess statements must contain a Monitor table index number that identifies which monitor table entry to use and a regular expression that identifies which process to monitor. The SystemEDGE agent will ignore the error and will continue to parse the configuration file.

nlist of /unix failed

The SystemEDGE agent was unable to obtain the kernel name list (or symbol table) for the UNIX operating system that is running on the underlying system. The agent will continue to operate but will be unable to report many kernel performance statistics and configuration parameters. This error occurs when an alternative kernel is booted (for example, /unix.boot) for testing purposes. To fix this problem, reboot with the /unix kernel file.

no extension variable found for X

The SystemEDGE agent was unable to find an extension variable that corresponds to index *x*. Extension variables are numbered from 1 through 32. This error message may occur if you attempt to add (through the configuration file) an extension variable whose number falls outside this range. The SystemEDGE agent will ignore the configuration file statement and will continue to parse the configuration file. For more information about extending the SystemEDGE agent, refer to Chapter 16, “Adding Custom MIB Objects.”

non-existent object to track history of

An *emphistory* configuration file statement contains an invalid object-identifier; that is, the object identifier does not exist within the SystemEDGE agent’s MIB. The agent will not create a History table entry and will ignore the statement. The agent will, however, continue to parse the remainder of the configuration file.

no process matching expression

The SystemEDGE agent was unable to match a configuration file *monprocess* regular expression to a corresponding process name. Consequently, it will ignore the *monprocess* statement in the configuration file.

not querying serial port status

The SystemEDGE agent will not query the status of serial ports in response to queries of the variable *hrDeviceStatus*. On some older UNIX systems, serial port status queries can interfere with serial-port based applications. The SystemEDGE agent configuration file parameter, *no_serial_status*, enables this option.

not sending authen failure traps

The SystemEDGE agent will not send MIB-II authenFailure trap messages in response to SNMP queries that are using invalid community strings. The SystemEDGE agent configuration file parameter, `no_authen_traps`, enables this option.

not stat'ing disks devices

The SystemEDGE agent will not check the status of disk devices according to the configuration file directive.

not stat'ng floppy devices

The SystemEDGE agent will not check the status of floppy disk devices according to the configuration file directive.

not stating NFS filesystems

The SystemEDGE agent will not monitor or report statistics for NFS-mounted filesystems. On some UNIX systems, attempts to ascertain the status of NFS-mounted filesystems whose file servers are unavailable or down can indefinitely block the SystemEDGE agent. Unfortunately, programmatic options to prevent this are not possible. The SystemEDGE agent configuration file parameter, `no_stat_nfs_filesystems`, enables this option.

not supporting actions

The SystemEDGE agent is not supporting actions according to the configuration file directive.

not supporting remoteShell group

The SystemEDGE agent will not support SNMP queries (Gets and Sets) to the remoteShell group because local system security policies prohibit this functionality. The SystemEDGE agent configuration file parameter, `no_remoteshell_group`, enables this option.

not supporting user/group tables

The SystemEDGE agent will not support SNMP queries to the User and Group tables because local system security policies prohibit the dissemination of valid user and group information. The SystemEDGE agent configuration file parameter, `no_usergroup_table`, enables this option.

not supporting who table

The SystemEDGE agent will not support SNMP queries to the Who table because local system security policies prohibit the dissemination of currently logged in users. The SystemEDGE agent configuration file parameter, `no_who_table`, enables this option.

nventmon entry X not ready

The SystemEDGE agent has sent an NT Event Monitor trap message that indicates that entry `x` is notReady. The SystemEDGE agent will no longer perform Windows event monitoring on behalf of the NT Event Monitor table entry `x`, but will continue to perform Windows event monitoring on all other entries. An NT Event Monitor table entry can become notReady when the SystemEDGE agent is unable to read the corresponding Windows event log file. This error occurs only when there are errors on the underlying Windows system.

odm_initialize failed

On AIX systems, the SystemEDGE agent uses an odm library for obtaining hardware and device information. This message indicates that initialization of that library failed. Consequently, it may not support device information.

openProcess failed on pid

On Windows, the SystemEDGE agent failed to open a process for statistics retrieval because the process may not be in existence anymore. The agent will continue to operate normally and will continue to support the process table.

openProcessToken failed on pid

On Windows, the SystemEDGE agent failed to open a process for statistics retrieval because the process may not be in existence anymore. The agent will continue to operate normally and will continue to support the process table.

openprom device not supported

Upon startup on many UNIX systems, the SystemEDGE agent attempts to discover the devices contained in the underlying system using whatever operating system support is available. This error messages indicates that the underlying system does not support the openprom device, which is used to determine system configuration and hardware information. Consequently, the SystemEDGE agent will attempt to determine the system's configuration without the operating system's help.

perfDiskObjects X != Num_Disks

The SystemEDGE agent found a number of disk objects that is not equivalent to the number of disks it found through other mechanisms. You can ignore this message.

realloc of mnt cache failed!

The SystemEDGE agent failed to reallocate space for its internal cache of mounted file systems. This error usually occurs when the system is extremely low on memory. The agent will continue to operate but may be unable to report file system statistics until more memory is available.

recvfrom failed

The SystemEDGE agent encountered an error when reading an SNMP request from the underlying UDP transport.

reload_process_table: open /proc failed

The SystemEDGE agent failed to open the /proc directory on a UNIX system; consequently, it cannot support the Process Monitor table.

reload_process_table: proc ioctl failed

The SystemEDGE agent failed to perform an I/O control operation (ioctl) on a particular process located in the /proc directory. The agent, therefore, cannot support process information for this particular process, but will perform nominally for other processes.

root device ptr failed, no openprom

Upon startup on many UNIX systems, the SystemEDGE agent attempts to discover the devices contained in the underlying system using whatever operating system support is available. This error messages indicates that the underlying system does not support the openprom device, which is used to determine system configuration and hardware information. Consequently, the agent will attempt to determine the system's configuration without the operating system's help.

sent SIGKILL to process X

On UNIX systems, the SystemEDGE agent sent a KILL signal to a process whose PID is *x*. This function is accomplished through SNMP Sets to the processKill variable in the Process Monitor table or to the hrRunStatus variable in the Host Resources hrSWRunTable.

sent signal X to process X

On UNIX systems, the SystemEDGE agent sent a signal to a process whose PID is *x*. This function is accomplished through SNMP Sets to the processKill variable in the Process Monitor table or to the hrRunStatus variable within the Host Resources hrSWRunTable. Any valid UNIX signal can be sent to a process.

setLogmonEntry: invalid set (logfile), row of status

To set Log Monitor table rows that have a status of notInService, you must perform SNMP Set operations made to those rows. This error indicates that the Set operation it references failed.

setMonEntry: bad size for OID val

An SNMP Set operation to the Log Monitor table contained an incorrect length for a particular object-identifier value. The Set operation referenced by this message failed.

setMonEntry: invalid oper

An SNMP Set operation to the Log Monitor table contained an improper operator type. The Set operation referenced by this message failed.

setMonEntry: invalid oper type

An SNMP Set operation to the Log Monitor table contained an improper operator type. The Set operation referenced by this message failed.

setMonEntry: invalid stype

An SNMP Set operation to the Log Monitor table contained an improper sample type. The Set operation referenced by this message failed.

setMonEntry: invalid type for OID

An SNMP Set operation to the Log Monitor table contained an improper object identifier type. The Set operation referenced by this message failed.

setMonEntry: invalid type for val

An SNMP Set operation to the Log Monitor table contained an improper value type. The Set operation referenced by this message failed.

setMonEntry: oid type invalid

An SNMP Set operation to the Log Monitor table contained an improper object-identifier type. The Set operation referenced by this message failed.

setMonEntry: stype type invalid

An SNMP Set operation to the Log Monitor table contained an improper sample type. The Set operation referenced by this message failed.

stat logfilename failed

The SystemEDGE agent could not determine file information for a particular log file that it is monitoring for a regular expression. Consequently, it sets the status of the corresponding Log Monitor table row to notReady.

stat of extension command file X failed

The SystemEDGE agent could not determine file information for the extension command that corresponds to extension variable *x*. Consequently, the agent will not support extension variable *x*. The agent will support all other valid extension variables. For more information about extending the SystemEDGE agent, refer to Chapter 16, “Adding Custom MIB Objects.”

stat of logmon action X failed

The SystemEDGE agent could not determine file information about an action command file that is used with a Log Monitor configuration file statement. Consequently, it sets the status of the corresponding Log Monitor table row to notReady.

stat of monfilesys action X failed

The SystemEDGE agent could not determine file information about an action command file that is used with a monfilesys configuration file statement. Consequently, it sets the status of the corresponding Monitor table row to notReady.

stat of monprocess action X failed

The SystemEDGE agent could not determine file information about an action command file that is used with a monprocess configuration file statement. Consequently, it sets the status of the corresponding Process Monitor table row to notReady.

stat of nteventmon action X failed

The SystemEDGE agent could not determine file information about an action command file that is used with a NT Event Monitor table configuration file statement. Consequently, it sets the status of the corresponding NT Event Monitor table row to notReady.

sysedge using port X, config file Y

Upon startup, UNIX versions of the SystemEDGE agent report which UDP port and configuration file they are using. This message is informational only and does not represent an error condition.

system call ret error X

The SystemEDGE agent was unable to execute a command that was specified as part of the remoteShell group functionality. This error can occur when the remoteShell function is invalid, or if the underlying system cannot create a sub-process.

This agent binary is compiled for X, not Y

Because the SystemEDGE agent is specific to the revision of the operating system on which it runs, it must often be compiled specifically for each operating system. This error message indicates that a version of the SystemEDGE agent for one version of the operating system was executed on a version that is not compatible with the one for which it was compiled. For example, on Solaris 2.x, you must use separate SystemEDGE agent binaries for versions 2.5.x, and 2.6, 2.7 (32-bit) and 2.7 (64-bit). For a list of binaries for each operating system, refer to Chapter 19, “Troubleshooting and Usage Suggestions.”

timeGetDevCaps failed, exiting

On Windows systems, the SystemEDGE agent was unable to get the system's timer resolution capabilities that are necessary for internal operation and self-monitoring of MIB objects. Consequently, the agent will not operate. This problem is most likely due to an error on the underlying Windows system. To resolve this problem, reboot the Windows system.

timeKillEvent failed

On Windows systems, the SystemEDGE agent delays its initialization in order to avoid potential race conditions that can be created by the order in which the registry and services are initialized. This error message indicates that the SystemEDGE agent could not stop its internal timer event from firing. The problem most likely is due to an error on the underlying Windows system. To resolve this problem, reboot the Windows system.

trap ipaddress/hostname X invalid

On UNIX systems, the `sysedge.cf` configuration file indicates to which hosts the SystemEDGE agent should send SNMP trap messages. This error indicates that one of the trap statements in the `sysedge.cf` configuration file specifies an incorrect hostname or IP address. The agent will ignore the offending trap statement, but will parse the rest of the configuration file.

turning off process table support

The SystemEDGE agent is disabling support of the process table according to the configuration file directive.

turning off sets to Empire process table

The SystemEDGE agent is disabling support of SNMP Sets to the process table according to the configuration file directive.

two processes with PID X

The SystemEDGE agent has discovered two separate, distinct processes with the same process identifier. Although this condition is technically impossible, the SystemEDGE agent still guards against it. This message indicates that the agent has discovered this situation and has properly accommodated it.

two software packages with same index

The SystemEDGE agent has discovered two software packages with the same index value. This condition can occur when local users have changed files in the system's software installation area, or if those files have been damaged. This message indicates that the SystemEDGE agent has discovered the situation and has properly accommodated it. Report the conditions that generated this message by sending an e-mail to support-concord@ca.com.

unable to open monitor file

The SystemEDGE agent was unable to open the monitor file that was specified as a command-line argument or was unable to open the default monitor file. Verify that the monitor file that is specified as part of the UNIX command line is specifying a valid monitor file or that the default monitor file is in the proper location. The default monitor file, `sysedge.mon`, is contained in the UNIX `/dir` directory or in the Windows `%SystemRoot%\system32\` directory.

unable to process acl for community X

This message indicates that the SystemEDGE agent was unable to parse an access control list specification as part of a community declaration in the `sysedge.cf` configuration file. The agent will ignore the offending access control list but will continue to support the corresponding community string declaration.

unknown HP CPU type

This message indicates that the underlying HP-UX system contains a processor type that is not known to the SystemEDGE agent. Report this message and the underlying processor type to Technical Support by sending e-mail to support-concord@ca.com.

unknown NT event log name

This message indicates that an invalid Windows event log name was specified, either through SNMP or through the nteventmon command-line tool. Valid Windows event log names are application, security, or system. The agent will not create the NT Event Monitor table entry.

unknown NT event type

This message indicates that an invalid Windows event type was specified, either through SNMP or through the command-line tool nteventmon. Valid Windows event types are error, warning, information, success, fail, or all. The NT event monitor entry will not be created.

unknown service start type

This message indicates that the SystemEDGE agent discovered a Windows service start type that it did not understand. Report this message to Technical Support by sending e-mail to support-concord@ca.com.

unknown system type

This message indicates that the SystemEDGE agent could not determine if the underlying Windows system is configured as a Windows Server or a Windows Workstation. Report this message to Technical Support by sending an e-mail to support-concord@ca.com.

username X not found, all subprograms will be disabled

This message indicates that the username that is specified in the sysedge.cf configuration file for subprogram execution does not exist. Consequently, all subprogram execution by SystemEDGE agent will be disabled.

Using config file

This message indicates that the SystemEDGE agent for Windows is using the specified configuration file. This message is informational only and does not indicate that an error has occurred.

Using monitor file

This message indicates that the SystemEDGE agent for Windows is using the specified monitor configuration file. This message is informational only and does not indicate that an error has occurred.

using old config file

This message indicates that the SystemEDGE agent is using a configuration file from a release earlier than Release 3.0. This message is informational in nature.

using old monitor file X; updates will be placed in Y

This message indicates that the SystemEDGE agent is reading a monitor configuration file (*x*) for a release that was earlier than Release 3.0 and that it will write updated, Release 4.0 monitor configuration files to the file *y*.

Command-line Utility Error Messages

This section lists the error messages that can occur when you are using the various command-line utilities that are included with SystemEDGE. These error messages are generally printed to standard error or standard output. Messages are sorted alphabetically by tool.

The utilities included in this section are as follows:

- edgemon
- edgewatch
- emphistory
- nteventmon
- sendtrap
- snmpget
- sysvariable
- walktree
- xtrapmon

edgemon Error Messages

This section describes error messages that may be generated by the edgemon utility.

bogus port number

This message indicates that the optional agent port number that was provided to the monitor command was invalid and either contained non-numeric symbols or was negative. The requested action, therefore, was not performed.

cant resolve address

The SystemEDGE agent could not resolve the address or hostname that was provided to the monitor command with a valid IP address, and therefore did not perform the requested action. Verify that the hostname that was provided has a valid address through tools like host or nslookup.

Couldn't find filesystem at target host

The file system name that was provided to edgemon was not present or mounted on the remote system. Consequently, the Monitor table entry was not created.

edgemon: internal error; please report

An internal SNMP library error has occurred within the program. Report this error to Technical Support by sending an e-mail to support-concord@ca.com.

GET failed

An error occurred while the edgemon utility was communicating with the SystemEDGE agent. The request may or may not have succeeded. Verify that the community string and target IP address (or system name) are correct.

GET_NEXT failed

An error occurred while retrieving and parsing the Systems Management MIB Mounted Device table, perhaps due to a dropped SNMP packet or a SystemEDGE agent error. The operation may or may not have succeeded. For more information, refer to the other error or information messages returned by edgemon.

monitor: WSASStartup failed

On Windows systems, the monitor command was unable to initialize the WinSock library and could not communicate with the targeted SystemEDGE agent. The requested action, therefore, was not performed.

Object is not Integer based

The MIB object that was specified for monitored through the SystemEDGE agent's Monitor table, was not an integer-based object, so no monitor table entry was created. The agent can perform self-monitoring only on integer-based MIB objects, including counters, gauges, timeticks, and integers.

Set to monitor table failed

This message indicates that edgemon failed to perform an attempted SNMP Set operation to create, delete, or modify a row in the SystemEDGE agent Monitor table. The requested action, therefore, was not performed. This error can be caused by an invalid community-string or a community-string without write access.

SNMP Operation failed

An error occurred while retrieving and parsing the Systems Management MIB Mounted Device table, perhaps due to a dropped SNMP packet or a SystemEDGE agent error. The operation may or may not have succeeded, depending on other error or information messages returned by edgemon.

unknown command

An unknown command was passed to edgemon, so no operation was performed.

edgemon Error Messages

This section describes error messages that may be generated by the edgemon utility.

AIX does not support procMonAttribute

The Process Monitor attribute that was passed to edgemon is not supported by the underlying AIX operating system (on which the SystemEDGE agent is running). The agent therefore does not create a Process Monitor table. For a list of Process Monitor attributes that are not supported by AIX, refer to the *SystemEDGE Agent Release Notes*.

Digital UNIX/Tru64 does not support procMonAttribute

The Process Monitoring attribute that was passed to edgewatch is not supported by the underlying Digital UNIX or Tru64 operating system (on which the SystemEDGE agent is running). The agent therefore does not create a Process Monitor table. For a list of Process Monitor attributes that are not supported by Digital UNIX or Tru64, refer to the *eHealth SystemEDGE Release Notes*.

edgewatch: internal error; please report

An internal SNMP library error has occurred within the program. Report this error to Technical Support by sending an e-mail to support-concord@ca.com.

edgewatch: invalid timeout value

The timeout value that was passed to edgewatch was invalid, so the utility did not perform the requested action. If a timeout value is set, it must be an integer that is greater than 0.

edgewatch: WSASStartup failed

On Windows systems, edgewatch was unable to initialize the WinSock library and could not communicate with the targeted agent. The agent, therefore, did not perform the requested action.

GET failed

edgewatch failed to get one of the request variables because of an error in the SNMP Get operation. Verify that the community string and the target system hostname (or IP address) are correct.

GET_NEXT failed

An error occurred while edgewatch was communicating with the SystemEDGE agent during an SNMP Get Next operation. Verify that the community string, target IP address, and system name are correct. The request may have succeeded.

invalid/unknown system type

The system type of the target SytemEDGE agent is unknown to edgewatch, so the utility did not perform the requested operation. This error can occur when an older version of the relevant command-line utility is used with a newer version of the agent ported to a system not originally supported at the time this utility was written. For an updated copy of the utility, contact eHealth Technical Support for an updated copy of this utility.

NT does not support procMonAttribute

The Process Monitor attribute that was passed to edgewatch is not supported by the underlying Windows operating system (on which the SystemEDGE agent is running). The agent therefore does not create a Process Monitor table. For a list of Process Monitor attributes that are not supported by Windows, refer to the *eHealth SytemEDGE Release Notes*.

procAlive is the only attribute which can be applied to a process group

The SystemEDGE agent can support process monitoring of groups of processes only if the particular attribute is procAlive. If another attribute was passed to edgewatch, no operation will be performed.

Returned system type is not integer!

edgewatch queries the SystemEDGE agent for its system type; this error message indicates that the resulting value was not of type integer. Report this error to Technical Support by sending an e-mail to support-concord@ca.com.

Set to Empire procmon table failed

edgewidth failed to Set or create an entry in the SystemEDGE agent's Process Monitor table. Verify that the community string and target hostname (or IP address) are correct and that the community string has read-write permission.

Set to logMonitorTable failed

edgewidth failed to Set or create an entry in the SystemEDGE agent's Log Monitor table. Verify that the community string and target hostname (or IP address) are correct and that the community string has read-write permission.

Set to ntEventMonTable failed

edgewidth failed to Set or create an entry in the SystemEDGE agent's NT Event Monitor table. Verify that the community string and target hostname (or IP address) are correct and that the community string has read-write permission.

Set to procMon table failed

edgewidth failed to Set or create an entry in the SystemEDGE agent's Process Monitor table. Verify that the community string and target hostname (or IP address) are correct and that the community string has read-write permission.

SNMP Operation failed

edgewidth failed to get one of the request variables because of an error in the SNMP operation. Verify that the community string and target hostname (or IP address) are correct.

target host not a Windows NT system

edgewidth attempted to perform an operation that the SystemEDGE agent supports only on Windows systems on a system that was not running Windows. Therefore, it did not perform the operation. For example, UNIX systems do not support the NT Event Monitor table, so operations to the NT

Event Monitor table that are targeted to SystemEDGE agents that are running on UNIX systems will generate this error message. Verify your target hostname or IP address to ensure you are targeting a Windows system.

unknown NT event log name

An invalid Windows event log name was passed to edgewatch for NT Event Monitor table manipulation, so no operation was performed. Verify that the event log name is valid on the target system.

unknown NT event type

An invalid Windows event type was passed to edgewatch for NT Event Monitor table manipulation, so no operation was performed. Verify that the event type name is valid on the target system.

emphistory Error Messages

This section describes error messages that may be generated by the emphistory utility.

bogus port number

The optional SystemEDGE agent port number that was provided to emphistory was invalid and either contained non-numeric symbols or was negative. The requested action, therefore, was not performed.

cant resolve address

The address or hostname that was provided to emphistory could not be resolved to a valid IP address, so the requested action was not performed. Verify that the provided hostname has a valid address through tools like host or nslookup.

emphistory: WSAStartup failed

On Windows systems, emphistory was unable to initialize the WinSock library and could not communicate with the targeted SystemEDGE agent. The requested action, therefore, was not performed.

emphistory: internal error; please report

An internal SNMP library error has occurred within the program. Report this error to Technical Support by sending an e-mail to support-concord@ca.com.

Set to empireHistoryCtrlTable failed

emphistory failed to perform an attempted SNMP Set operation to create, delete, or modify a row in the SystemEDGE agent History Control table. The requested action, therefore, was not performed. This error can occur because of an invalid community-string or a community-string without write access.

unknown command

The requested command, passed to emphistory on the command-line, was invalid, so no action was performed.

nteventmon Error Messages

This section describes error messages that may be generated by the nteventmon utility.

cant resolve address for

The address or hostname provided to nteventmon could not be resolved to a valid IP address, so the requested action was not performed. Verify that the hostname has a valid address through tools like `host` or `nslookup`.

GET failed

nventmon first attempts to verify that the target machine is running Windows. This message indicates that the SNMP query that was used in that verification process failed, and that no NT Event Monitor table entry was created. The SNMP query may fail if the corresponding packets are dropped or if the target agent is not a SystemEDGE agent.

nventmon: internal error; please report

An internal SNMP library error has occurred within the program. Report this error to Technical Support by sending an e-mail to support-concord@ca.com.

nventmon: unknown NT event log name

The Windows event log name that was passed to nventmon was invalid. Valid event log names include the following: application, security, and system. Try the command again with a valid Windows event log name.

nventmon: unknown NT event type

The Windows event type password to nventmon was invalid. Valid Windows event types include the following: error, warning, info, information, succ, success, fail, failure, or all. Try the command again with a valid Windows event type.

nventmon: WSASStartup failed

On Windows systems, nventmon was unable to initialize the WinSock library and could not communicate with the targeted SystemEDGE agent. The requested action, therefore, was not performed.

Returned system type is not integer!

nventmon first attempts to verify that the target machine is running Windows. This message indicates that the SNMP query that was used in that verification process returned a non-integer value. Therefore, no NT Event Monitor table entry was created.

Set to ntEventMonTable failed

This message indicates that nteventmon failed to perform an attempted SNMP Set operation to create, delete, or modify a row in the SystemEDGE agent NT Event Monitor table. The requested action, therefore, was not performed. This error can result from an invalid community-string or a community-string without write access.

SNMP Operation failed

An nteventmon SNMP operation failed. This error may or may not indicate that nteventmon failed to perform its intended action; that depends on any other messages that are displayed by nteventmon.

target host is not an NT system

nteventmon first attempts to verify that the target machine is running Windows. This message indicates that the target system is not running Windows and therefore does not support the NT Event Monitor table.

unknown command

The command that was passed to nteventmon on the command-line was invalid. No action, therefore, was performed.

sendtrap Error Messages

This section describes error messages that may be generated by the sendtrap utility.

cant resolve address for

The address or hostname that was provided to sendtrap could not be resolved to a valid IP address, so the requested action was not performed. Verify that the hostname has a valid address through tools like host or nslookup.

error converting type, skipping

Variable bindings can be passed to sendtrap and then sent in an SNMP Trap PDU. This message indicates that the variable type that was specified for one of the variable bindings was invalid. An SNMP Trap PDU was sent, but the corresponding variable binding was not included.

error converting value, skipping

Variable bindings can be passed to sendtrap and then sent in an SNMP Trap PDU. This message indicates that the variable type that was specified for one of the variable bindings was invalid. An SNMP Trap PDU was sent, but the corresponding variable binding was not included.

error: bad specific-trap type

The specific-trap type that was specified to sendtrap was invalid, so an SNMP Trap PDU was not sent. Specific-trap types must be greater than or equal to 0.

error: bad trap type

The trap type that was specified to sendtrap was invalid, so an SNMP Trap PDU was not sent. Trap types must be greater than or equal to 0.

error parsing arguments, ignoring line

The sendtrap utility encountered an error parsing a variable binding. The utility ignored the offending line and sent the SNMP Trap PDU.

no more varbinds supported

The maximum number of variable bindings was already read and placed in the SNMP Trap PDU to be sent. All current and subsequent variable bindings will be ignored. The last SNMP Trap PDU was, however, sent.

sendtrap: WSASStartup failed

On Windows systems, sendtrap was unable to initialize the WinSock library and could not send the SNMP trap message.

snmpsend failed

The underlying library routine for sending the SNMP Trap PDU failed, and no trap PDU was sent.

warning: bogus SNMPv1 Trap type, still sending

SNMPv1 trap must be of types 1 through 6. This message indicates that the specified trap type was outside the valid range. The trap type was included in the SNMP Trap PDU, and it was still sent.

snmpget Error Messages

This section describes error messages that may be generated by the snmpget utility.

cant resolve address for

The address or hostname that was provided to snmpget could not be resolved to a valid IP address, so the requested action was not performed. Verify that the hostname has a valid address through tools like host or nslookup.

GET_NEXT failed

An error occurred while snmpget was communicating with the SystemEDGE agent. Verify that the community string and target IP address or system name are correct. The request may or may not have succeeded.

invalid timeout value

The timeout value passed to snmpget was invalid. The timeout value, if set, should be an integer number greater than 0. The requested action, therefore, was not performed.

snmpget: returned object of type instead of

snmpget was expecting one object-type, but the SystemEDGE agent returned another type. Verify that the object-identifier that was passed to snmpget is correct.

SNMP Get failed

snmpget failed to get one of the request variables due to an error in the SNMP Get operation. Verify that the community string and target IP address (or system name) are correct. The request may or may not have succeeded.

snmpget: WSASStartup failed

On Windows systems, snmpget was unable to initialize the WinSock library and could not communicate with the targeted SystemEDGE agent. The requested action, therefore, was not performed.

sysvariable Error Messages

This section describes error messages that may be generated by the sysvariable utility.

cant resolve address for

The address or hostname that was provided to sysvariable could not be resolved to a valid IP address, so the requested action was not performed. Verify that the hostname has a valid address through tools like host or nslookup.

GET failed

sysvariable failed to get one of the request variables due to an error in the SNMP Get operation. Verify that the community string is valid and that the target system hostname (or IP address) are correct.

SNMP Operation failed

sysvariable failed to get one of the request variables due to an error in the SNMP operation. Verify that the community string and target hostname (or IP address) are correct.

sysvariable: internal error; please report

An internal SNMP library error has occurred within the program. Report this message to Technical Support by sending an e-mail to support-concord@ca.com.

sysvariable: WSASStartup failed

On Windows systems, sysvariable was unable to initialize the WinSock library and could not communicate with the targeted SystemEDGE agent. The requested action, therefore, was not performed.

walktree Error Messages

This section describes error messages that may be generated by the walktree utility.

cant resolve address for

The address or hostname that was provided to walktree could not be resolved to a valid IP address, so the requested action was not performed. Verify that the hostname has a valid address through tools like host or nslookup.

ERROR: could not create output file

walktree could not open the specified output file due to an error opening or creating it. No walk operation was performed.

walktree: internal error; please report

An internal SNMP library error has occurred within the program. Report this error to Technical Support by sending an e-mail to support-concord@ca.com.

walktree: retries should be at least 1

walktree attempts to retry failed SNMP requests because SNMP packets may be dropped or lost. This error message indicates that the retry value that was specified on the command line was invalid.

walktree: WSASStartup failed

On Windows systems, walktree was unable to initialize the WinSock library and could not communicate with the targeted agent. The requested action, therefore, was not performed.

xtrapmon Error Messages

This section describes error messages that may be generated by the xtrapmon utility.

Parse error for trap from

xtrapmon encountered an error parsing a trap from an SNMP agent.

Received PDU of type X instead of Trap

xtrapmon received an SNMP PDU that was not a trap. Normally, xtrapmon receives only Trap PDUs.

xtrapmon: failed to get trap socket

xtrapmon failed to bind to port UDP/162, the SNMP trap port. This problem usually occurs if another process is already bound to that port. You can solve this problem by terminating the other trap-receiving process.

xtrapmon: snmprecv failed

xtrapmon encountered an error in the library routine that receives and parses SNMP PDUs. Therefore, it did not return a Trap PDU.

xtrapmon: WSASStartup failed

On Windows systems, xtrapmon was unable to initialize the WinSock library and could therefore not bind to the SNMP trap port. It will therefore not receive any traps.

Using the syslog Facility

The SystemEDGE agent uses the UNIX syslog facility to log informational messages and error conditions that it may encounter during its operation. For more information about the syslog facility, refer to the following man pages for more information on the syslog facility:

- `syslog(3)`
- `syslog.conf(5)`
- `syslogd(8)`

You can also edit the `sysedge.cf` file to instruct the SystemEDGE agent to log these messages in a different facility. For instructions, refer to “Configuring Alternative Syslog Facilities (UNIX Only)” on page 134.

Logging syslog Messages

syslog messages are typically logged to the `/var/adm/messages` or `/usr/adm/syslog` text file (depending on your system and how the syslog daemon is configured on that system). By default, the SystemEDGE agent daemon uses syslog to log messages with priority levels of *informational* through *emergency*. If you are running the agent in debug mode with the runtime command line option `-d`, syslog will also log messages of the debug priority level.

This guide does not provide a complete tutorial on the syslog utility. Instead, it describes how to configure the syslog daemon on your system to log messages from daemon processes like the SystemEDGE agent to a text file (/var/log/daemon-log for Sun SPARC systems, and /usr/adm/daemon-log for HP-UX systems).

Every message that is sent using syslog includes a **facility code** that tells where the message is coming from, and a **priority code** that tells the severity of the message. Table 84 describes the available syslog facility codes.

Table 84. syslog Facility Codes

Facility Code	Description
LOG_KERN	Kernel messages
LOG_USER	Random user-level messages
LOG_MAIL	Mail system
LOG_DAEMON	System daemons
LOG_AUTH	Security/authorization messages
LOG_SYSLOG	Messages generated internally by syslog
LOG_LPR	Line printer subsystem
LOG_NEWS	Network News subsystem
LOG_UUCP	UUCP subsystem
LOG_CRON	Cron/at subsystem

Table 85 describes the syslog priority codes.

Table 85. syslog Priority Codes

Priority Code	Description
LOG_EMERG	System is unusable.
LOG_ALERT	Immediate action is required.
LOG_CRIT	Critical condition.
LOG_ERR	Error condition.
LOG_WARNING	Warning condition.
LOG_NOTICE	Normal but signification condition.
LOG_INFO	Informational.
LOG_DEBUG	Debug-level messages.

The typical syslog configuration logs messages to the text file `/var/adm/messages` on Sun SPARC systems and to `/usr/adm/syslog` on HP-UX Release 10.x and 11.x systems. If the message has a priority of `LOG_ERR` or higher, syslog also displays the message to the console.

Creating a Log File for Daemon Messages

Because the typical log file contains messages from many facilities in addition to daemon processes like the SystemEDGE agent, you may want to configure all messages from daemon processes to be logged to a separate daemon-log file. If your `/etc/syslog.conf` file does not already contain an entry for logging daemon processes separately, you can add an entry to cause syslog to log those messages to a separate daemon log file.

Creating a Daemon Log File for Sun SPARC Systems

To create a separate file for logging daemon messages for a Sun SPARC system, edit the `/etc/syslog.conf` file by entering following:

```
daemon /var/log/daemon-log
```

Creating a Daemon Log File for HP-UX Systems

To create a separate file for logging daemon messages for HP-UX (versions 10.x and 11.x) systems, edit the `/etc/syslog.conf` file by entering following:

```
daemon.info /usr/adm/daemon-log
```

These changes take effect when the system is rebooted or when the syslog daemon reads its configuration file again. You can set it to do so by sending it an HUP signal.

Adding Self-Monitoring Entries to the sysedge.mon File

This appendix describes the format for the `sysedge.mon` file, which provides stable storage for the self-monitoring and history tables supported by SystemEDGE agent. In most cases, you do not need to edit the `sysedge.mon` file. Instead, you can configure these features of the SystemEDGE agent through the `sysedge.cf` configuration file or through one of the command-line utilities. If you do need to edit `sysedge.mon`, use this appendix to ensure that you are using the correct file format.

SystemEDGE Table Backing Store

On startup, the SystemEDGE agent's `sysedge.mon` file reports to the agent the state of the self-monitoring tables while the agent was previously running. The agent looks for `sysedge.mon` in the `/etc` or `%SystemRoot%\system32\` directories by default when it is started, unless you have specified an alternate directory and file name. This monitor configuration file consists of a series of entries, each describing a row in one of the self-monitoring tables. For descriptions of valid values and more examples, refer to Chapter 10, "Configuring Threshold Monitoring."

NOTE

Edit the sysedge.mon file *only* when SystemEDGE agent is not running. The SystemEDGE agent *overwrites* this file every time the stored tables (Monitor, Process Monitor, Process Group Monitor, Log Monitor, NT Event Monitor, or History Control) are modified.

Before you edit this file, copy it from the SystemEDGE agent distribution to the /etc directory or the system root directory, as follows.

To copy sysedge.mon to the /etc directory on a UNIX system, enter the following command:

```
cp config/sysedge.mon /etc
```

To copy sysedge.mon to the system root directory on a Windows system, enter the following command at the command prompt:

```
copy config\sysedge.mon %SystemRoot%\system32\
```

This file consists of comments and table entries. Lines that are empty or begin with a pound sign (#) are treated as comments and are ignored. Comments conclude at the end of the each line.

Adding Monitor Table Entries to the sysedge.mon File

Monitor table entries begin with the keyword monitor. They are delineated by open and closed brackets, and include ten fields. Table 86 describes the fields of the Monitor table entries for the sysedge.mon file.

Table 86. Fields for Monitor Table Entries in sysedge.mon (Page 1 of 3)

Field Name	Description
monitor {	Beginning of the entry is marked by an open bracket ({}).
monIndex	<p>Integer (1 to MAXINT) that indicates the row index for this entry. Rows 1 through 10 are reserved for the agent's internal use; the index for additional rows must fall in the range of 11 to MAXINT.</p> <p>The index is particularly important because SNMP does not directly support creation and deletion of MIB objects; instead, it creates and deletes them as side effects of SNMP Set operations. This limitation means that the person creating or modifying Monitor table entries through the monitor command or by editing sysedge.mon must know the exact MIB table index to use for row creation.</p>
monDescr	Quoted string that is 0 to 128 characters in length and that normally contains a description of the object that is being monitored, as well as a severity level for this event.
monInterval	<p>Integer value (30 to MAXINT) that indicates how often (in seconds) the monitoring should be performed. For example, the value 30 instructs the agent to monitor this entry every 30 seconds.</p> <p>NOTE: This value must be a multiple of 30 seconds.</p>
monSampleType	Either absoluteValue(1) or deltaValue(2); this value indicates whether this entry should sample the object's absolute value or whether the agent should take the difference between successive samples. For example, when monitoring counter variables, use deltaValue because it describes the rate of change. When monitoring gauges, use absoluteValue because it describes the object's exact value.
monOID	<p>Complete object-instance identifier that represents the value to be monitored.</p> <p>Note that the instance portion of the object-identifier (for example, .0 for scalars) is also required. The object-instance must exist and must be contained within the SystemEDGE agent's supported MIBs. That is, any supported (integer-based) object that exists in MIB-II, the Host Resources MIB, or the Systems Management MIB is valid. Objects should be of integer type, including counter, gauge, integer, or enumerated integer.</p>

Table 86. Fields for Monitor Table Entries in sysedge.mon (Page 2 of 3)

Field Name	Description
monOperator	<p>The operator type is a boolean operator used for evaluating the expression: <i>currval operator value</i></p> <p>The operator can be one of the following:</p> <ul style="list-style-type: none"> • nop (no operation; monitor the object's value, but do not evaluate the Boolean expression) • gt (greater than) • lt (less than) • ge (greater than or equal to) • le (less than or equal to) • eq (equal) • ne (not equal)
monValue	<p>Integer value to which the current value of the monitored MIB variable is compared during each monitoring cycle. If the comparison evaluates to true, (where the operator-type tells how to compare them), the agent sends a trap. For example, if you wanted to be notified if the value of some gauge goes over 100, you would set 100 as the monValue against which the current value of the gauge is compared.</p>
monRowStatus	<p>One of the following:</p> <ul style="list-style-type: none"> • active • notInService • notReady • createAndGo • createAndWait <p>Normally, a row is either active or notInService. These values are identical in meaning to the SNMPv2 SMI RowStatus textual convention. For more information, refer to Appendix E.</p>
monAction	<p>Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when the expression evaluates to true and a trap is sent. If the string is empty, no action will be performed for this entry.</p>

Table 86. Fields for Monitor Table Entries in sysedge.mon (Page 3 of 3)

Field Name	Description
monFlags	Unsigned integer flags value that indicates additional behavioral semantics that this row should follow during the course of its operation. By default, this field is assigned the hexadecimal value 0x00. For more information about this field, refer to Chapter 10, “Configuring Threshold Monitoring.”
}	End of the entry is marked by a closed bracket ({}).

Sample Monitor Table Entries in sysedge.mon

This section includes examples for adding entries to the Monitor table through the sysedge.mon file.

Monitoring 1-Minute Load Average

The following entry in the sysedge.mon file instructs the SystemEDGE agent to monitor the 1 minute load average every 60 seconds. If the sampled value is greater than 300 (3.00 in this example because SNMP does not support real numbers), the agent sends an SNMP trap message.

```
monentry {
  11
  "Monitor 1 minute load average"
  60
  absoluteValue
  1.3.6.1.4.1.1.546.1.1.7.8.26.0
  gt
  300
  active
  " "
  0x0
}
```

Monitoring File Systems

The following entry in the `sysedge.mon` file instructs the SystemEDGE agent to monitor how full the `/` file system is every 2 minutes (120 seconds). If the file system becomes 95% full, the agent sends an SNMP trap message.

```
monentry {  
    19  
    "Monitor / filesystem"  
    120  
    absoluteValue  
    1.3.6.1.4.1.546.1.1.1.7.1.14.1  
    ge  
    95  
    active  
    ""  
    0x0  
}
```

NOTE

The object-instance identifier for the SystemEDGE agent may not be the same across all instantiations of the agent.

Adding Process Monitor Table Entries to the sysedge.mon File

Process Monitor table entries begin with the keyword `processmon`. They are delineated by open and closed brackets and include eleven fields. Table 87 describes the fields for Process Monitor table entries in the `sysedge.mon` file.

Table 87. Fields for Process Monitor Table Entries (Page 1 of 3)

Field Name	Description
<code>processmon {</code>	Beginning of the entry is marked by an open bracket (<code>{</code>).
<code>pmonIndex</code>	Integer (1 to MAXINT) that indicates the row index for this entry. The index is particularly important because SNMP does not directly support creation and deletion of MIB objects; instead, it creates and deletes them as side effects of SNMP Set operations. This limitation means that the person creating or modifying Process Monitor table entries through the <code>processmon</code> command or by editing <code>sysedge.mon</code> must know the exact MIB table index to use for row creation.
<code>pmonDescr</code>	Quoted string that is 0 through 128 characters in length and that normally contains a description of the process and attribute that the agent is monitoring, as well as a severity level for this event.
<code>pmonInterval</code>	Integer value (30 to MAXINT) that indicates how often (in seconds) the agent should perform this monitoring. For example, the value 30 instructs the agent to monitor this entry every 30 seconds. NOTE: This value must be a multiple of 30 seconds.
<code>pmonSampleType</code>	Either <code>absoluteValue(1)</code> or <code>deltaValue(2)</code> ; this value indicates whether the agent should sample the attribute's absolute value or take the difference between successive samples. For example, use <code>deltaValue</code> to monitor counter attributes because it provides the rate of change. Use <code>absoluteValue</code> to monitor gauges, because it provides the object's exact value.

Table 87. Fields for Process Monitor Table Entries (Page 2 of 3)

Field Name	Description
pmonAttribute	Process attribute that is being monitored. For a complete list of supported attributes, refer to Chapter 11, “Configuring Process and Service Monitoring.” For example, to monitor a process to ensure that it is alive, specify the attribute procAlive. To track the number of packets received by the particular application or process, specify procMsgsSent.
pmonOperator	Operator type; A boolean operator used for evaluating the following expression: <code>currval operator value</code> The operator can be one of the following: <ul style="list-style-type: none"> • nop (no operation; monitor the object’s value, but do not evaluate the Boolean expression) • gt (greater than) • lt (less than) • ge (greater than or equal to) • le (less than or equal to) • eq (equal) • ne (not equal)
pmonValue	Integer value to which the current value of the monitored process attribute is compared during each monitoring cycle. If the comparison evaluates to True, (where the operator type tells how to compare them), the agent sends a trap. For example, if you want to be notified if the value of a gauge goes above 100, set 100 as the pmonValue against which the current value of the gauge is compared.
pmonFlags	Integer flags value that indicates additional behavioral semantics this row should follow during the course of its operation. By default, this field is assigned the hexadecimal value 0x00. For more information about this field, refer to Chapter 11, “Configuring Process and Service Monitoring.”

Table 87. Fields for Process Monitor Table Entries (Page 3 of 3)

Field Name	Description
pmonAction	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when the expression evaluates to True and a trap is sent. If the string is empty, no action will be performed for this entry.
pmonRowStatus	<p>One of the following:</p> <ul style="list-style-type: none"> • active • notInService • notReady • createAndGo, • createAndWait <p>Normally, a row is either active or notInService. These values are identical in meaning to the SNMPv2 SMI RowStatus textual convention. For more information, refer to Appendix E.</p>
}	End of the entry is marked by a closed bracket (}).

Sample Process Monitor Entries in sysedge.mon

This section includes examples for adding entries to the Process Monitor table through the sysedge.mon file.

Monitoring the Netscape Process Run Status

The following entry in the sysedge.mon file instructs the SystemEDGE agent to monitor the netscape process every 60 seconds. If the process is down, the agent sends an SNMP trap message.

```
processmon {  
    1  
    "Monitor netscape alive"  
    60  
    absoluteValue  
    procAlive  
    eq  
    4  
    0x0  
    " "  
    "netscape"  
    active  
}
```

Monitoring the Netscape Process Size

The following entry in the sysedge.mon file instructs the SystemEDGE agent to monitor the memory utilization of the netscape process every 2 minutes (120 seconds). If the process RSS value exceeds 50000, a SNMP Trap message is sent.

```
processmon {  
    2  
    "Monitor netscape RSS"  
    60  
    absoluteValue  
    procRSS  
    gt  
    50000  
    0x0  
    " "  
    "netscape"  
    active  
}
```

Adding Process Group Monitor Table Entries to the sysedge.mon File

Process Group Monitor table entries begin with the keyword `procgroupmon`. They are delineated by open and closed brackets. Table 88 describes the fields for Process Group Monitor table entries in the `sysedge.mon` file.

Table 88. Fields for Process Group Monitor Table Entries (Page 1 of 2)

Field Name	Description
<code>procgroupmon {</code>	Beginning of the entry is marked by an open bracket (<code>{</code>).
<code>pgmonIndex</code>	Integer (1 to MAXINT) that indicates the row index for this entry. The index is particularly important because SNMP does not directly support creation and deletion of MIB objects; instead, it creates and deletes them as side effects of SNMP Set operations. This limitation means that the person creating or modifying Process Monitor table entries through the <code>processmon</code> command or by editing <code>sysedge.mon</code> must know the exact MIB table index to use for row creation.
<code>pgmonDescr</code>	Quoted string that is 0 through 128 characters in length and that normally contains a description of the process and attribute that the agent is monitoring, as well as a severity level for this event.
<code>pgmonInterval</code>	Integer value (30 to MAXINT) that indicates how often (in seconds) the agent should perform this monitoring. For example, the value 30 instructs the agent to monitor this entry every 30 seconds. NOTE: This value must be a multiple of 30 seconds.
<code>pgmonProcRegExpr</code>	Regular expression to apply when the agent is attempting to match processes by name.
<code>pgmonFlags</code>	Integer flags value that indicates additional behavioral semantics this row should follow during the course of its operation. By default, this field is assigned the hexadecimal value 0x00. For more information about this field, refer to Chapter 12, “Configuring Process Group Monitoring.”

Table 88. Fields for Process Group Monitor Table Entries (Page 2 of 2)

Field Name	Description
pgmonAction	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when the expression evaluates to True and a trap is sent. If the string is empty, no action will be performed for this entry.
pgmonRowStatus	<p>One of the following:</p> <ul style="list-style-type: none">• active• notInService• notReady• createAndGo,• createAndWait <p>Normally, a row is either active or notInService. These values are identical in meaning to the SNMPv2 SMI RowStatus textual convention. For more information, refer to Appendix E.</p>
}	End of the entry is marked by a closed bracket (}).

Sample Process Group Monitor Entry in sysedge.mon

This section includes an example for adding entries to the Process Group Monitor table through the sysedge.mon file.

Monitoring the httpd Process Group

The following entry in the sysedge.mon file instructs the SystemEDGE agent to monitor the httpd process group every 60 seconds.

```
procgrouppmon {  
    1  
    "Monitor Web process group"  
    60  
    'httpd'  
    0x0  
    ""  
    active  
}
```

Adding Log Monitor Table Entries to the sysedge.mon File

Log Monitor table entries begin with the keyword `logmon`. They are delineated by open and closed brackets, and they include seven fields. Table 89 describes the field for Log Monitor table entries in the `sysedge.mon` file.

Table 89. Fields for Log Monitor Table Entries (Page 1 of 2)

Field Name	Description
<code>logmon {</code>	Beginning of the entry is marked by an open bracket (<code>{</code>).
<code>logMonitorIndex</code>	Identifies the row of the table. The index is particularly important because SNMP does not directly support creation and deletion of MIB objects; instead, it creates and deletes them as side effects of SNMP Set operations. This limitation means that the person creating or modifying Log Monitor table entries through the <code>watch logfile</code> command or by editing <code>sysedge.mon</code> must know the exact MIB table index to use for row creation.
<code>logMonitorLogFile</code>	Complete path and file name of the log file to be monitored.
<code>logMonitorRegularExpression</code>	Regular expression to apply when scanning the log file for matches. For information about the rules for specifying regular expressions, refer to the UNIX man page on <code>egrep(1)</code> .
<code>logMonitorAction</code>	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command, along with any parameters, to be executed when the regular expression is matched and a trap is sent. If the string is empty, no action will be performed for this entry.
<code>logMonitorFlags</code>	Unsigned integer flags value indicating additional behavioral semantics this row should follow during the course of its operation. By default, this field is assigned the hexadecimal value <code>0x00</code> . For more information about this field, refer to Chapter 14, “Configuring Windows Event Monitoring.”

Table 89. Fields for Log Monitor Table Entries (Page 2 of 2)

Field Name	Description
logMonitorRowStatus	<p>One of the following:</p> <ul style="list-style-type: none"> • active • notInService • notReady • createAndGo, • createAndWait <p>Normally, a row is either active or notInService. These values are identical in meaning to the SNMPv2 SMI RowStatus textual convention. For more information, refer to Appendix E.</p>
}	End of the entry is marked by a closed bracket (}).

Sample Log Monitor Entry in sysedge.mon

This section includes an example for adding an entry to the Log Monitor table through the sysedge.mon file.

Monitoring for Failed su Attempts

The following entry instructs the SystemEDGE agent to monitor the /var/adm/messages log file for the expression su.*fail. If a match is found, the agent sends an SNMP trap message.

```
logmon {
    2
    "SU - WARNING"
    "/var/adm/messages"
    "su.*fail"
    ""
    0x0
    active
}
```

Adding NT Event Monitor Table Entries to the sysedge.mon File

NT Event Monitor table entries begin with the keyword `nventmon`. They are delineated by open and closed brackets, and they include seven fields. Table 90 describes the field for NT Event Monitor table entries in the `sysedge.mon` file.

Table 90. Fields for NT Event Monitor Entries (Page 1 of 2)

Field Name	Description
<code>nventmon {</code>	Beginning of the entry is marked by an open bracket (<code>{</code>).
<code>ntEventMonIndex</code>	Identifies the row of the table. The index is particularly important because SNMP does not directly support creation and deletion of MIB objects; instead, it creates and deletes them as side effects of SNMP Set operations. This limitation means that the person creating or modifying NT Event Monitor table entries through the <code>watch nvent</code> command or by editing <code>sysedge.mon</code> must know the exact MIB table index to use for row creation.
<code>ntEventMonDescription</code>	Quoted text string that contains a description of the purpose, function, and (optionally) creator of the entry.
<code>ntEventMonLog</code>	Integer that designates which Windows Event Log to monitor. The following are possible values: <ul style="list-style-type: none">• Application(1)• Security(2)• System(3)

Table 90. Fields for NT Event Monitor Entries (Page 2 of 2)

Field Name	Description
ntEventMonTypeFilter	Number that identifies the event type to match for this entry. Types 1 through 5 are defined by Windows as follows: <ul style="list-style-type: none"> • error(1) • warning(2) • information(3) • success(4) • failure(5) Type all(6) indicates that all event types should match.
ntEventMonSrcFilter	Regular expression to apply to the Event Source when scanning the events for matches. For more information about specifying regular expressions, refer to the UNIX man page on egrep(1).
ntEventMonDescFilter	Regular expression to apply to the Event Description when scanning the events for matches. For more information about specifying regular expressions, refer to the UNIX man page on egrep(1).
ntEventMonStatus	SNMPv2 RowStatus; one of the following: <ul style="list-style-type: none"> • active(1) • notInService(2) • notReady(3)
ntEventMonAction	Quoted string that is 0 to 256 characters in length and that specifies the full path of the command (along with any parameters) to be executed when a match is found and a trap is sent. If the string is empty, no action will be performed for this entry.
ntEventMonFlags	Unsigned integer flags value indicating additional behavioral semantics this row should follow during the course of its operation. By default, this field is assigned the hexadecimal value 0x00. For more information about this field, refer to Chapter 15, “Configuring History Collection.”
}	End of the entry is marked by a closed bracket (}).

Sample NT Event Monitor Entries in sysedge.mon

This section includes an example for adding an entry to the NT Event Monitor table through the sysedge.mon file.

Monitoring for Application Errors

The following entry instructs the SystemEDGE agent to monitor the Application NT Event Log for events of type Error. If a match is found, the agent sends an SNMP trap message.

```
ntheventmon {  
    5  
    "Application - ERROR"  
    Application  
    Error  
    ".*"   
    ".*"   
    active  
    ""  
    0x0  
}
```

Adding History Control Table Entries to the sysedge.mon File

History Control Table entries begin with the keyword `history`. They are delineated by open and closed brackets, and they include six fields. Table 91 describes the field for NT Event Monitor table entries in the `sysedge.mon` file.

Table 91. Columns of the History Control Table (Page 1 of 2)

Column Name	Description
<code>history {</code>	Beginning of the entry is marked by an open bracket (<code>{</code>).
<code>empireHistoryControlIndex</code>	Integer (1 to MAXINT) that uniquely identifies the entry in the table. The index is particularly important because SNMP does not directly support creation and deletion of MIB objects; instead, it creates and deletes them as side effects of SNMP Set operations. This limitation means that the person creating or modifying History Control table entries through the <code>emphistory</code> command or by editing <code>sysedge.mon</code> must know the exact MIB table index to use for row creation.
<code>empireHistoryControlDescr</code>	Text string that describes the data-collection function defined by this entry, and that may include who created it.
<code>empireHistoryControlBuckets</code>	Total number of samples to be stored for this variable.
<code>emphistoryControlObjID</code>	Complete object-instance identifier of the MIB variable to be sampled. Note that you must include the <i>instance</i> portion of the object-identifier (for example, <code>.0</code> for scalars). The object-instance must exist and must be contained within the Systems Management MIB. For example, any supported (integer-based) object in MIB-II, the Host Resources MIB, or the Systems Management MIB is valid. Objects should be of integer type including counter, gauge, integer, or enumerated integer.
<code>empHistoryControlInterval</code>	Integer value indicating how often (in seconds) to sample the MIB variable. The interval <i>must</i> be a multiple of 30 seconds.

Table 91. Columns of the History Control Table (Page 2 of 2)

Column Name	Description
empHistoryControlStatus	<p>One of the following:</p> <ul style="list-style-type: none"> • active • notInService • notReady • createAndGo • createAndWait <p>These values are identical in meaning to the SNMPv2 SMI RowStatus textual convention. For more information, refer to Appendix E.</p> <p>Setting the status to destroy(6) causes the agent to discontinue history sampling for this entry, and to delete both this row and the corresponding data sample rows in the empireHistoryTable.</p>
}	End of the entry is marked by a closed bracket (}).

Sample History Control Table Entries in sysedge.mon

This section includes an example for adding an entry to the History Control table through the sysedge.mon file.

Disk Transfer History

The following entry instructs the SystemEDGE agent to collect the disk transfer statistics for the first physical disk. This is entry index 10. It will keep 100 samples and take a new sample every 60 seconds.

```
history {
  10
  "Disk 1 Transfers"
  100
  1.3.6.1.4.1.546.12.1.1.6.1
  60
  active
}
```


Textual Conventions for Row Status

This appendix provides information on the SNMPv2 textual conventions for row status. This entire appendix is directly excerpted from RFC 1443, *Textual Conventions for SNMPv2* (Case et al., 1993).

RFC 1443: Textual Conventions for SNMPv2

RowStatus ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

The RowStatus textual convention is used to manage the creation and deletion of conceptual rows, and is used as the value of the SYNTAX clause for the status column of a conceptual row (as described in Section 7.7.1 of [2].)

The status column has six defined values:

- active, which indicates that the conceptual row is available for use by the managed device.
- notInService, which indicates that the conceptual row exists in the agent, but is unavailable for use by the managed device.
- notReady, which indicates that the conceptual row exists in the agent, but is missing information necessary in order to be available for use by the managed device.

- `createAndGo`, which is supplied by a management station wishing to create a new instance of a conceptual row and to have it available for use by the managed device.
- `createAndWait`, which is supplied by a management station wishing to create a new instance of a conceptual row but not to have it available for use by the managed device.
- `destroy`, which is supplied by a management station wishing to delete all of the instances associated with an existing conceptual row.

Whereas five of the six values (all except `notReady`) may be specified in a management protocol set operation, only three values will be returned in response to a management protocol retrieval operation: `notReady`, `notInService` or `active`. That is, when queried, an existing conceptual row has only three states: it is either available for use by the managed device (the status column has value `active`); it is not available for use by the managed device, though the agent has sufficient information to make it so (the status column has value `notInService`); or, it is not available for use by the managed device, because the agent lacks sufficient information (the status column has value `notReady`).

NOTE

This textual convention may be used for a MIB table, irrespective of whether the values of that table's conceptual rows are able to be modified while it is active, or whether its conceptual rows must be taken out of service in order to be modified. That is, it is the responsibility of the DESCRIPTION clause of the status column to specify whether the status column must be `notInService` in order for the value of some other column of the same conceptual row to be modified.

To summarize the effect of having a conceptual row with a status column having a SYNTAX clause value of RowStatus, consider the following state diagram.

ACTION	STATE			
	A	B	C	D
	status column does not exist	status column is notReady	status column is notInService	status column is active
set status column to createAndGo	noError → D or inconsistentValue	inconsistentValue	inconsistentValue	inconsistentValue
set status column to createAndWait	noError (refer to Note 1) or wrongValue	inconsistentValue	inconsistentValue	inconsistentValue
set status column to active	inconsistentValue	inconsistentValue Refer to Note 2 → D	noError → D	noError → D
set status column to notInService	inconsistentValue	inconsistentValue Refer to Note 3 → C	noError	noError → C or wrongValue
set status column to destroy	noError → A	noError → A	noError → A	noError → A
set any other column to some value	Refer to Note 4. → A	noError Refer to Note 1.	noError → C	noError → D

(1) goto B or C, depending on information available to the agent.

(2) if other variable bindings included in the same PDU, provide values for all columns which are missing but required, then return noError and goto D.

(3) if other variable bindings included in the same PDU, provide values for all columns which are missing but required, then return `noError` and goto C.

(4) at the discretion of the agent, either `noError` or `inconsistentValue` may be returned.

NOTE

Other processing of the set request may result in a response other than `noError` being returned, e.g., `wrongValue`, `noCreation`, etc.

Conceptual Row Creation

There are four potential interactions when creating a conceptual row: selecting an instance-identifier which is not in use; creating the conceptual row; initializing any objects for which the agent does not supply a default; and, making the conceptual row available for use by the managed device.

Interaction 1: Selecting an Instance-Identifier

The algorithm used to select an instance- identifier varies for each conceptual row. In some cases, the instance-identifier is semantically significant, e.g., the destination address of a route, and a management station selects the instance-identifier according to the semantics.

In other cases, the instance-identifier is used solely to distinguish conceptual rows, and a management station without specific knowledge of the conceptual row might examine the instances present in order to determine an unused instance- identifier. (This approach may be used, but it is often highly sub-optimal; however, it is also a questionable practice for a naive management station to attempt conceptual row creation.)

Alternately, the MIB module which defines the conceptual row might provide one or more objects which provide assistance in determining an unused instance-identifier. For example, if the conceptual row is indexed by an integer-value, then an object having an integer-valued SYNTAX clause might be defined for such a purpose, allowing a management station to issue a management protocol retrieval operation. In order to avoid unnecessary collisions between competing management stations, adjacent retrievals of this object should be different.

Finally, the management station could select a pseudo-random number to use as the index. In the event that this index was already in use and an inconsistentValue was returned in response to the management protocol set operation, the management station should simply select a new pseudo-random number and retry the operation.

A MIB designer should choose between the two latter algorithms based on the size of the table (and therefore the efficiency of each algorithm). For tables in which a large number of entries are expected, it is recommended that a MIB object be defined that returns an acceptable index for creation. For tables with small numbers of entries, it is recommended that the latter pseudo-random index mechanism be used.

Interaction 2: Creating the Conceptual Row

Once an unused instance-identifier has been selected, the management station determines if it wishes to create and activate the conceptual row in one transaction or in a negotiated set of interactions.

Interaction 2a: Creating and Activating the Conceptual Row

The management station must first determine the column requirements, i.e., it must determine those columns for which it must or must not provide values. Depending on the complexity of the table and the management station's knowledge of the agent's capabilities, this determination can be made locally by

the management station. Alternately, the management station issues a management protocol get operation to examine all columns in the conceptual row that it wishes to create. In response, for each column, there are three possible outcomes:

- A value is returned, indicating that some other management station has already created this conceptual row. We return to interaction 1.
- The exception `noSuchInstance` is returned, indicating that the agent implements the object-type associated with this column, and that this column in at least one conceptual row would be accessible in the MIB view used by the retrieval were it to exist. For those columns to which the agent provides read- create access, the `noSuchInstance` exception tells the management station that it should supply a value for this column when the conceptual row is to be created.
- The exception `noSuchObject` is returned, indicating that the agent does not implement the object-type associated with this column or that there is no conceptual row for which this column would be accessible in the MIB view used by the retrieval. As such, the management station can not issue any management protocol set operations to create an instance of this column.

Once the column requirements have been determined, a management protocol set operation is accordingly issued. This operation also sets the new instance of the status column to `createAndGo`.

When the agent processes the set operation, it verifies that it has sufficient information to make the conceptual row available for use by the managed device. The information available to the agent is provided by two sources: the management protocol set operation which creates the conceptual row, and, implementation-specific defaults supplied by the agent (note that an agent must provide implementation-specific defaults for at least those objects which it implements as read-only). If there is sufficient information available, then the conceptual row is created, a `noError` response is returned, the status

column is set to active, and no further interactions are necessary (i.e., interactions 3 and 4 are skipped). If there is insufficient information, then the conceptual row is not created, and the set operation fails with an error of `inconsistentValue`. On this error, the management station can issue a management protocol retrieval operation to determine if this was because it failed to specify a value for a required column, or, because the selected instance of the status column already existed. In the latter case, we return to interaction 1. In the former case, the management station can re-issue the set operation with the additional information, or begin interaction 2 again using `createAndWait` in order to negotiate creation of the conceptual row.

NOTE

Regardless of the method used to determine the column requirements, it is possible that the management station might deem a column necessary when, in fact, the agent will not allow that particular columnar instance to be created or written. In this case, the management protocol set operation will fail with an error such as `noCreation` or `notWriteable`. In this case, the management station decides whether it needs to be able to set a value for that particular columnar instance. If not, the management station re-issues the management protocol set operation, but without setting a value for that particular columnar instance; otherwise, the management station aborts the row creation algorithm.

Interaction 2b: Negotiating the Creation of the Conceptual Row

The management station issues a management protocol set operation which sets the desired instance of the status column to `createAndWait`. If the agent is unwilling to process a request of this sort, the set operation fails with an error of `wrongValue`. (As a consequence, such an agent must be prepared to accept a single management protocol set operation, i.e., interaction 2a

above, containing all of the columns indicated by its column requirements.) Otherwise, the conceptual row is created, a `noError` response is returned, and the status column is immediately set to either `notInService` or `notReady`, depending on whether it has sufficient information to make the conceptual row available for use by the managed device. If there is sufficient information available, then the status column is set to `notInService`; otherwise, if there is insufficient information, then the status column is set to `notReady`. Regardless, we proceed to interaction 3.

Interaction 3: Initializing Non-defaulted Objects

The management station must now determine the column requirements. It issues a management protocol get operation to examine all columns in the created conceptual row. In the response, for each column, there are three possible outcomes:

- A value is returned, indicating that the agent implements the object-type associated with this column and had sufficient information to provide a value. For those columns to which the agent provides read-create access, a value return tells the management station that it may issue additional management protocol set operations, if it desires, in order to change the value associated with this column.
- The exception `noSuchInstance` is returned, indicating that the agent implements the object-type associated with this column, and that this column in at least one conceptual row would be accessible in the MIB view used by the retrieval were it to exist. However, the agent does not have sufficient information to provide a value, and until a value is provided, the conceptual row may not be made available for use by the managed device. For those columns to which the agent provides read-create access, the `noSuchInstance` exception tells the management station that it must issue additional management protocol set operations, in order to provide a value associated with this column.

- The exception `noSuchObject` is returned, indicating that the agent does not implement the object-type associated with this column or that there is no conceptual row for which this column would be accessible in the MIB view used by the retrieval. As such, the management station can not issue any management protocol set operations to create an instance of this column.

If the value associated with the status column is `notReady`, then the management station must first deal with all `noSuchInstance` columns, if any. Having done so, the value of the status column becomes `notInService`, and we proceed to interaction 4.

Interaction 4: Making the Conceptual Row Available

Once the management station is satisfied with the values associated with the columns of the conceptual row, it issues a management protocol set operation to set the status column to active. If the agent has sufficient information to make the conceptual row available for use by the managed device, the management protocol set operation succeeds (a `noError` response is returned). Otherwise, the management protocol set operation fails with an error of `inconsistentValue`.

NOTE

A conceptual row having a status column with value `notInService` or `notReady` is unavailable to the managed device. As such, it is possible for the managed device to create its own instances during the time between the management protocol set operation which sets the status column to `createAndWait` and the management protocol set operation which sets the status column to active. In this case, when the management protocol set operation is issued to set the status column to active, the values held in the agent supersede those used by the managed device.

If the management station is prevented from setting the status column to active (e.g., due to management station or network failure) the conceptual row will be left in the 'notInService' or notReady state, consuming resources indefinitely. The agent must detect conceptual rows that have been in either state for an abnormally long period of time and remove them. This period of time should be long enough to allow for human response time (including think time) between the creation of the conceptual row and the setting of the status to active. It is suggested that this period be approximately 5 minutes in length.

Conceptual Row Suspension

When a conceptual row is active, the management station may issue a management protocol set operation which sets the instance of the status column to notInService. If the agent is unwilling to do so, the set operation fails with an error of wrongValue. Otherwise, the conceptual row is taken out of service, and a noError response is returned. It is the responsibility of the DESCRIPTION clause of the status column to indicate under what circumstances the status column should be taken out of service (e.g., in order for the value of some other column of the same conceptual row to be modified).

Conceptual Row Deletion

For deletion of conceptual rows, a management protocol set operation is issued which sets the instance of the status column to destroy. This request may be made regardless of the current value of the status column (e.g., it is possible to delete conceptual rows which are either notReady, notInService or active.) If the operation succeeds, then all instances associated with the conceptual row are immediately removed.

SYNTAX

INTEGER {

The following two values are states; these values may be read or written:

active(1)

notInService(2)

The following value is a state; this value may be read, but not written:

notReady(3)

The following three values are actions; these values may be written, but are never read:

createAndGo(4)

createAndWait(5)

destroy(6)

}

Bibliography

This appendix lists related works that may be helpful when you are using the SystemEDGE agent.

Bach, M. J. (1986). *The Design of the UNIX Operating System*. Prentice Hall.

Carpenter, G. and Wijnen, B. (1991). SNMP-DPI - Simple Network Management Protocol Distributed Program Interface. Request for Comments (Proposed Standard) RFC 1228, Internet Engineering Task Force. (Obsoleted by RFC1592).

Case, J. D., Fedor, M., Schoffstall, M. L., and Davin, C. (1990). Simple Network Management Protocol. Anonymous FTP. RFC 1157 (Obsoletes RFC 1098).

Case, J., McCloghrie, K., Rose, M., and Waldbusser, S. (1993). Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2). Anonymous FTP. RFC 1443.

Cockcroft, A. (1995). *Sun Performance and Tuning: Sparc & Solaris*. SunSoft Press.

Friedl, J.E.F. (1997). *Mastering Regular Expressions: Powerful Techniques for Perl and Other Tools*. O'Reilly. 1st edition.

Gookin, D. (1993). *Batch Files and Beyond: Your Path to PC Power!* McGraw Hill, 4th edition.

Grillo, P. and Waldbusser, S. (1993). Host Resources MIB. Anonymous FTP. RFC 1514.

Krupczak, B. (1993). UNIX Systems Management via SNMP. In *Proceedings of the IFIP TC6/WG6.6 Third International Symposium on Integrated Network Management*.

Krupczak, B. (1995). Systems Management and the Internet Management Framework, *ConneXions: The Interoperability Report*, 9(8): 2-9.

Leffler, S. J., McKusick, M. K., Karels, M. J., and Quateman, J. S. (1989). *The Design and Implementation of the 4.3 BSD UNIX Operating System*. Addison-Wesley 2nd edition.

Mauro, D. and Schmidt, K. (2001). *Essential SNMP*. O'Reilly.

McCloghrie, K. and Rose, M. T. (1991). Management Information Base for network management of TCP/IP-based internets: MIB-II. Anonymous FTP. RFC 1213 (Obsoletes RFC 1158).

Microsoft (1995a). *Microsoft Windows NT Networking Guide*. Microsoft Press.

Microsoft (1995b). *Microsoft Windows NT Resource Kit*. Microsoft Press.

Rose, M. (1991a). SNMP MUX Protocol and MIB. Request for Comments (Proposed Standard) RFC 1227, Internet Engineering Task Force.

Rose, M. and McCloghrie, K. (1990). Structure and Identification of Management Information for TCP/IP-based Internets. Anonymous FTP. RFC 1155 (Obsoletes RFC 1065).

Rose, M. T. (1991b). *The Simple Book: An Introduction to Management of TCP/IP-Based Internets*. Prentice Hall, 1st edition.

Sun Microsystems, Inc. (1988). RPC: Remote Procedure Call Protocol Specification: Version 2. Anonymous FTP. RFC 1057.

Sun Microsystems, Inc. (1989). NFS: Network File System Protocol Specification. Anonymous FTP. RFC 1094.

Sun Microsystems, Inc. (1990). *SunOS Reference Manual*. Sun Microsystems, Vol II edition.

Wijnen, B., Carpenter, G., Curran, K., Sehgal, A., and Waters, G. (1994). Simple Network Management Protocol Distributed Protocol Interface Version 2.0. Request for Comments (Proposed Standard) RFC 1592, Internet Engineering Task Force (Obsoletes RFC1228).

Index

A

access

communities

configuring 128

default 130

setting 49

lists, specifying 129

actions

configuring support 137

default parameters

Log Monitor table 340

Monitor table 255

NT Event Monitor table 366

Process Group Monitor table 326

Process Monitor table 296

error messages 439

Log Monitor table 341

Monitor table 256

NT Event Monitor table 366

overview 35

Process Group Monitor table 326

Process Monitor table 296

adding

custom MIB objects 34, 391

entries

History Control table 389, 511

Log Monitor table 348, 506

Monitor table 494

NT Event Monitor table 508

Process Group Monitor table 329

Process Monitor table 499, 503

performance registry variables 405

addrChangeTrap 228

AdvantEDGE View

configuring

History Control table 384

Log Monitor table 342

Monitor table 256

NT Event Monitor table 368

Process Monitor table 298, 327

using with SystemEDGE 37

agent multiplexing 158

AIMs

configuring support 143

enabling in sysedge.cf 143

overview 39

See also eHealth application insight
modules.

AIX

installing SystemEDGE 82

platform support 23

removing SystemEDGE 102

- SNMP agent 163
 - system requirements 82
- asset tracking 34
- assigning entry rows
 - Monitor table 257
 - Process Group Monitor table 327
 - Process Monitor table 298
- audience 24
- authentication failure traps
 - UNIX 132
- auto.install file 96
- automating
 - deployment of SystemEDGE agent
 - configuring software for distributed systems 419
 - methods 417
 - security issues 420
 - installation 96

B

- bibliography 525
- bin subdirectory
 - all platforms 96
 - UNIX 90
 - Windows 92
- Boot Configuration group, Systems Management MIB 169
- bounce.exe 92

C

- call-sendtrap.c file 96
- cfile.txt 93
- changing process nice value 175
- checkfile.exe 90, 92
- checkprinter.exe 96
- CIM SNMP agent 164

- collecting history
 - History Control table 206
 - history sampling 379
 - History table 205
 - overview 33
- config subdirectory 94
- configuration files
 - overview 86
 - sysedge.mon 259
- configuring
 - access communities 128
 - actions 137
 - agent debugging support 135
 - AIM support 143
 - alternative syslog facilities
 - UNIX 134
 - Windows 134
 - authentication failure traps
 - UNIX 132
 - disk probing 136
 - during installation 118
 - extension variables 393
 - history collection support 142
 - History Control table
 - AdvantEDGE View 384
 - dynamically 386
 - emphistory directive 385
 - methods 384
 - Linux free memory support 145
 - log file monitoring 141
 - Log Monitor table
 - AdvantEDGE View 342
 - dynamically 345
 - edgewatch utility 345
 - methods 343
 - Monitor table
 - AdvantEDGE View 256

- dynamically 261
- methods 258
- sysedge.mon file 259
- NT Event Monitor table
 - AdvantEDGE View 368
 - dynamically 369
 - methods 368
 - watch ntevent directive 369
- overview 117
- permissions for subprograms 142
- Process Group Monitor table
 - dynamically 328
 - methods 328
 - overview 141
 - watch procgroup directive 329
- Process Monitor table
 - AdvantEDGE View 298, 327
 - dynamically 299
 - methods 298
 - support 139
 - watch process directive 300
- Remote Shell group support 133
- security 146
- status checking
 - floppy devices 135
 - serial ports 136
- sysedge.cf file 118
- system information
 - UNIX 128
- threshold monitoring 245
- Top Processes AIM 144
- trap communities 130
- User and Group support 133
- Who Table support 132
- Windows
 - event monitoring 357
 - registry and performance variables 408

- copying sysedge.cf 119
- countmail.sh 96
- CPU Statistics group 211
- creating log file for daemon messages 491

D

- daemon messages, logging 491
- debugext.sh 399
- debugging SystemEDGE agent 135
- deleting
 - entries
 - History Control table 389
 - Log Monitor table 349
 - Monitor table 275
 - Process Group Monitor table 332
 - Process Monitor table 311
 - interprocess communication 181
- deploying SystemEDGE agent 413
- diagsysedge.exe 94, 423, 426
- Digital UNIX
 - installing SystemEDGE 84
 - platform support 23
 - removing SystemEDGE 103
 - SNMP agent 163
 - system requirements 84
- disabling
 - automatic reinitialization of agent 291
 - entries
 - History Control table 389
 - Log Monitor table 349
 - NT Event Monitor table 376
 - event logging 291
 - support for remote file system checking 138
- disk
 - performance statistics, enabling on Windows 208
 - probing, configuring support 136

- Disk Statistics group 208
- Disk Storage Table 239
- distributing software through protocols 417
- documentation conventions 25
- dynamic configuration
 - History Control table 386
 - Log Monitor table 345
 - Monitor table 261
 - NT Event Monitor table 369
 - Process Group Monitor table 328
 - Process Monitor table 299

E

- edgemon utility
 - directory 90
 - error messages 472
 - overview 274
 - removing entries
 - Log Monitor table 350
 - Monitor table 279
 - NT Event Monitor table 377
 - Process Group Monitor table 332
 - Process Monitor table 316
- edgemon.1 man page 91
- edgemon.exe 93
- edgemon.txt 93
- edgemon utility
 - arguments 312
 - command arguments 346
 - directory 90
 - error messages 474
 - examples 313, 348
 - introduction 300
 - parameters 310, 346
 - syntax 310, 345
 - using 345
- edgemon.1 man page 91
- edgemon.exe 93
- edgemon.txt 93
- eHealth 40
- eHealth application insight modules 39
- eHealth for Cisco CallManager 38
- eHealth Service Availability 37
- eHealth Voice Quality Monitor 38
- email.exe 90, 93
- emphistory directive
 - example 386
 - parameters 386
 - syntax 385
- emphistory utility
 - description 90
 - error messages 478
 - Man page 91
 - overview 387
 - parameters 387
- emphistory.exe 93
- emphistory.txt 94
- empire.asn.1 97
- enable disk performance statistics on
 - Windows 208
- enterprise system object identifier 215
- error messages
 - actions 439
 - authentication failure 439
 - command-line utilities 472
 - descriptions 439
 - edgemon utility 472
 - edgemon utility 474
 - emphistory utility 478
 - invalid PID 440
 - nteventmon utility 479
 - sendtrap utility 481
 - snmpget utility 483

- sysvariable utility 484
- xtrapmon utility 486
- event logs
 - criteria for searching 359
 - monitoring 360
- examples
 - edgewatch utility 313, 348
 - extending Systems Management MIB 396
 - Extension group 396
 - history monitoring 389
 - log file monitoring 348
 - ntRegPerf group 409
 - process group monitoring 330
 - process monitoring 306
 - threshold monitoring 264
 - watch logfile 344
 - watch ntevent 371
 - Windows event monitoring 371, 375
- executing remote commands 177
- extending Systems Management MIB
 - adding custom MIB objects 391
 - Extension group 212
 - ntRegPerf group 403
- extension directive
 - parameters 394
 - specifying additional parameters 394
 - syntax 393
- Extension group
 - examples
 - pinging remote system 397
 - returning DNS domain name 396
 - returning NIS domain name 396
 - features 392
 - sample MIB branch 392
 - Systems Management MIB 212, 391
 - writing extension scripts 397
- extension scripts, writing 397
- extension variables

- configuring 393
- editing empire.asn1 400
- editing separate MIB specification 400
- examples 392
- pinging remote system 397
- returning DNS domain name 396
- returning NIS domain name 396
- using with management software 399

F

- Fault Manager 42
- file system space, monitoring 168
- flags
 - Log Monitor table 337
 - Monitor table 252
 - NT Event Monitor table 363
 - Process Group Monitor table 325
 - Process Monitor table 289
- floppy devices, configuring status checking 135

G

- getextension.sh 96
- getver.exe 93
- Group table 173

H

- history collection
 - collecting disk transfer history 512
 - History Control table 206
 - History table 205
 - introduction 33
 - overview 379
- History Control table
 - adding entries 511
 - AdvantEDGE View display 384

- columns 381
- configuring 384
- configuring support 142
- configuring with emphistory directive 385
- description 381
- emphistory utility 387
- examples
 - adding entries 389
 - collecting disk transfer history 512
 - deleting entries 389
 - listing entries 389
 - retrieving stored data samples 390
 - setting status of entries 389
- overview 380
- sample sysedge.mon entries 512
- Systems Management MIB 205

history sampling 379

History table

- columns 382
- description 382
- Systems Management MIB 205

Host Resources MIB

- Device group 236
- Disk Storage table 239
- File System table 240
- introduction 29
- overview 233
- Partition table 239
- Processor table 238
- Running Software group 241
- Storage group 236
- System group 234

Host System group 166

hostmib.asn1 97

HP-UX

- installing SystemEDGE 75

- platform support 23
- removing SystemEDGE 101
- SNMP agent 162
- system requirements 75

I

I/O Buffer Cache group, Systems Management MIB 184

identifying processes that consume the most CPU 33

implementing trap severity levels 436

indicating

- change in IP address 228
- change in process group 229
- Log Monitor entry is notReady 219
- match in log file 218
- match in Windows event log file 220
- Monitor event 216
- monRowsStatus is notReady 217
- previous condition no longer exists 222
- process attribute is no longer at threshold 226
- process attributed has reached threshold 225
- process has restarted 224
- process has stopped running 223
- status of NT Event Monitor entry is notReady 221

installation

- directories 90
- files 90
- subdirectories
 - bin 90, 92
 - config 94
 - doc 91, 93, 97

installing SystemEDGE

- AIX 82
- Digital UNIX 84
- HP-UX 75
- Linux 80
- Solaris 52
- Tru64 84
- Windows 59
- interprocess communication
 - deleting 181
 - tracking 180

K

- Kernel
 - Configuration group, Systems Management MIB 168
 - Performance group, Systems Management MIB 178

L

- license key
 - email request 110
 - example 106
- license utility 109
- licenseutil.pl utility 112
- licensing
 - e-mail request 110
 - license key 106
 - license utility 110
 - licenseutil.pl utility 112
 - obtaining a license key 108
 - overview 105
 - public key 105
 - troubleshooting 429
 - Web-based form 111
- Linux
 - installing SystemEDGE 80
 - platform support 23

- removing SystemEDGE 102
- system requirements 80
- Linux free memory
 - configuring support 145
 - enabling in sysedge 145
- listing entries
 - History Control table 389
 - Log Monitor table 348
- Live Health 42
- log files, monitoring 141, 333
- Log Monitor table
 - actions 341
 - adding entries, sysedge.mon 506
 - AdvantEDGE View display 342
 - columns 335
 - configuring 343
 - default action parameters 340
 - description 335
 - examples
 - adding an entry 348
 - deleting entries 349
 - disabling entries 349
 - listing entries 348
 - monitoring log files 507
 - searching for pop connection attempts 344
 - searching for su attempts 345
 - flags 337
 - monitoring log files 141
 - overview 204, 333
 - removing entries 349
 - sample sysedge.mon entries 507
 - Systems Management MIB 204
 - watch logfile directive 343
- logfileaction.sh file 96
- logging
 - action commands 263
 - agent-operating messages 155

- daemon messages 491
- syslog messages 489
- logmon keyword 506

M

- mailaction.sh file 96
- Message Buffer Allocation table, Systems Management MIB 182
- message buffers, monitoring 182
- MIB
 - Host Resources 29
 - MIB II 28
 - specification 97
 - supported by SystemEDGE agent 28
 - Systems Management 29
- monitor directive
 - monitor keyword 494
 - parameters 262
 - usage 261
- Monitor table
 - actions 256
 - adding entries through sysedge.mon 494
 - AdvantEDGE View display 256
 - assigning entry rows 257
 - columns 247
 - configuring 258
 - default action parameters 255
 - description 247
 - disabling traps 254
 - edgemon utility 274
 - examples
 - monitoring /usr file system 272
 - monitoring 15-minute load average 266
 - monitoring 1-minute load average 264, 497

- monitoring 5-minute load average 265
- monitoring input packet rate 270
- monitoring interrupt rate 267
- monitoring number of processes 274
- monitoring output packet rate 270
- monitoring page-fault rate 268
- monitoring root file system 271
- monitoring SNMP packets received 271
- flags
 - definitions 254
 - overview 252
- monitor directive
 - sysedge.cf 259
- overview 202
- removing entries 279
- sample sysedge.mon entries 497
- threshold monitoring 245
- monitoring
 - /usr file system 272
 - 15-minute load average 266
 - 1-minute load average 264
 - 1-minute load averagesysedge.mon 497
 - 5-minute load average 265
 - boot configuration 169
 - call quality 38
 - CPU statistics 211
 - devices 236
 - disk I/O statistics 208
 - disk-storage devices 239
 - DT processes 331
 - file systems 168, 240
 - groups of processes 203, 319
 - host system 234
 - httpd process, monitoring example in sysedge.mon 505

- I/O buffers 184
- input packet rate 270
- interrupt rate 267
- kernel configuration 168
- kernel performance 178
- log files
 - edgewidth utility 345
 - overview 333
 - sample Log Monitor table 204
 - sysedge.mon example 507
- message buffers 182
- mounted devices 167
- netscape process
 - edgewidth 314
 - sysedge.mon 502
- NFS facilities 186
- number of processes 274
- output packet rate 270
- page-fault rate 268
- partitions for disk-storage devices 239
- process
 - groups with watch procgroup 329
 - size 309
 - status 302
- process attributes 287
- processes 203, 281
- processors 238
- queues 180
- root file system 271
- RPC facilities 185
- running
 - processes 174
 - software 241
- semaphores 180
- sendmail 307
- services 281
- shared memory 180
- Simple TCP/IP Services service 308

- size of netscape process 502
- SNMP packets received 271
- storage areas 236
- streams 183
- Streams subsystem 170
- system information 166
- TCPVCS process 307
- thresholds 202
- user
 - account information 172
 - groups 173
 - logged on to system 176
- voice quality 38
- Windows
 - cache performance 194
 - event logs 358, 510
 - events 197
 - memory performance 195
 - page file performance 197
 - performance extensions 199
 - registry 191, 405
 - services 192
 - system 188
 - system performance 193
 - threads 190
- Xterm processes 330
- ypbind process
 - edgewidth 313
 - watch procAlive 308
- monolithic agents 159
- Mounted Devices table 167
- mqcnt file 96
- multiple SNMP agents 157

N

- NFS group 186
- nhAddSysEdgeMonEntries command 40
- nice value, changing 175

- NT Cache Performance group 194
 - NT Event Monitor group 197
 - NT Event Monitor table
 - actions 366
 - adding entries 508
 - AdvantEDGE View display 368
 - columns 360
 - configuring 368
 - default action parameters 366
 - description 360
 - event monitoring 357
 - examples
 - monitoring log files 510
 - searching application log for specific events 372
 - searching application log for Web server messages 371
 - searching security log for failure events 371
 - flags 363
 - overview 141
 - removing entries 376
 - sample sysedge.mon entries 510
 - watch ntevent directive 369
 - NT Memory Performance group 195
 - NT Page File Performance group 197
 - NT Registry and Performance Extension group 199
 - NT Registry group 191
 - NT Service group 192
 - NT System group 188
 - NT System Performance group 193
 - NT Thread group 190
 - ntdist.pl 96
 - nteventmon utility
 - error messages 479
 - nteventmon keyword 508
 - ntRegPerf directive
 - parameters 408
 - syntax 408
 - ntRegPerf group
 - examples
 - returning number of transmitted TCP segments 410
 - returning path to dump file 409
 - returning total number of current threads 410
 - overview 403
 - sample MIB group 404
 - unsupported Windows performance data types 407
 - valid variable types 404
 - ntRegPerf variables
 - editing empire.asn1 411
 - editing separate MIB specification 411
 - using with management software 410
- ## O
- obtaining
 - diagnostic information about the agent 423
 - troubleshooting data 423
 - obtaining a license key 108
 - operating system
 - patches 437
 - operating systems supported 23
 - optimizing row creation
 - History table 383
 - Log Monitor table 337
 - Monitor table 251
 - NT Event Monitor table 363
 - Process Group Monitor table 324
 - Process Monitor table 289

P

- patches required 437
- perfmon extensions 34
- ping.sh file 97
- platforms supported 23
- private-enterprise traps 215
- procAlive attribute 302
- process attributes 287
- Process Group Monitor table
 - actions 326
 - assigning entry rows 327
 - columns 320
 - configuring support 141
 - default action parameters 326
 - description 320
 - examples
 - monitoring DT process group 331
 - monitoring emacs process group 330
 - monitoring httpd process 505
 - monitoring xterm process group 330
 - flags 325
 - overview 203, 319
 - removing entries 331
 - sample sysedge.mon entries 505
 - watch procgroup directive 329
- process group monitoring
 - configuring support 141
 - introduction 32
- Process Monitor table
 - actions
 - default parameters 296
 - disabling 291
 - overview 297
 - adding entries through sysedge.mon 499, 503
 - AdvantEDGE View display 298, 327
 - assigning entry rows 298
 - columns 284
 - configuration examples 306
 - configuring support 139
 - description 284
 - disabling traps 291
 - examples
 - monitoring netscape process 502
 - monitoring process size 309
 - monitoring sendmail 307
 - monitoring Simple TCP/IP Services service 308
 - monitoring size of netscape process 502
 - monitoring TCPSVCS process 307
 - monitoring ypbind 308
 - flags
 - description 291
 - overview 289
 - overview 283
 - process attributes 287
 - processClear trap 294
 - removing entries 315
 - sample sysedge.mon entries 501
 - sample table entry 282
 - Systems Management MIB 203
 - using edgewatch utility 310
- process monitoring
 - introduction 32
 - sample entry 282
 - sending signal to a process 176
- Process table 174
- processClear trap, example 294
- processes, automatically restarting 436
- processmon keyword 499
- procGroupChangeTrap 229
- procgroupmon keyword 503

R

- remote file system, status checking 138
- Remote Shell group
 - configuring support 133
 - Systems Management MIB 177
- removing
 - entries
 - History Control table 389
 - Log Monitor table 349
 - Monitor table 279
 - NT Event Monitor table 376
 - Process Group Monitor table 331
 - Process Monitor table 315
- SystemEDGE
 - AIX 102
 - Digital UNIX 103
 - HP-UX 101
 - Linux 102
 - Solaris 98
 - Tru64 103
 - Windows 99
- reserving rows in self-monitoring tables 258
- restarting processes 436
- restartproc.sh 90
- restartsvc.exe 93, 97
- retrieving stored data samples 390
- row status
 - active 513
 - createAndGo 514
 - createAndWait 514
 - destroy 514
 - notInService 513
 - notReady 513
- RPC group 185

S

- S990sysedge script 95
- S99sysedge script 95
- searching
 - application log
 - specific events 372
 - Web server messages 371
 - event logs, criteria 359
 - logs
 - pop connection attempts 344
 - su attempts 345
 - security log for failure events 371
- security issues
 - deploying the agent 420
 - action commands 421
 - extension variables 421
 - MIBs 421
- self-monitoring 31
- sending signal to a process 176
- sendpage.pl 97
- sendtrap utility
 - directory 90
 - error messages 481
- sendtrap.1 Man page 91
- sendtrap.exe 92
- sendtrap.txt 94
- serial ports, configuring status checking 136
- service monitoring 32
- setting
 - policy for monitoring tables 257
 - status of History Control table entries 389
- signal, sending to a process 176
- smtp-listenq.sh 97
- SNMP
 - access communities 49
 - communities 48
 - deleting rows 522

- message types 48
- overview 47
- requests, troubleshooting 425
- row creation 516
- row suspension 522
- supporting multiple agents 157
- trap
 - communities 50
 - severity levels 436
 - troubleshooting 428
- snmpget utility, error messages 483
- Solaris
 - installing SystemEDGE 52
 - platform support 23
 - removing SystemEDGE 98
 - starting SystemEDGE 152
 - system requirements 52
- Solstice Enterprise Agent 160
- specifying
 - access list 129
 - MIB objects to monitor 259
- starting SystemEDGE
 - automatically 151
 - AIX 154
 - Digital UNIX 155
 - HP-UX 153
 - Linux 154
 - Solaris 152
 - Tru64 155
 - command line 150
 - overview 149
- starting Windows Master agent 153
- stopping Windows Master agent 150
- Streams
 - Buffer Allocation table, Systems Management MIB 183
 - group, Systems Management MIB 170
- subprogram support, configuring 142
- supported platforms 23
- supporting
 - custom MIB objects 34
 - multiple SNMP agents
 - agent multiplexing 158
 - AIX SNMP agent 163
 - CIM SNMP agent 164
 - Digital UNIX SNMP agent 163
 - HP-UX SNMP agent 162
 - Microsoft Windows Extensible agent 161
 - monolithic agents 159
 - overview 157
 - Solstice Enterprise Agent 160
 - Tru64 SNMP agent 163
- sysedge
 - file 90
 - script 95
- sysedge.1 man page 91
- sysedge.cf
 - directory 94
 - sample file 119
- sysedge.cf.5 man page 91
- sysedge.cpl 92
- sysedge.dll 92
- sysedge.exe 92
- sysedge.lic 86
 - directory 94
 - sample file 106
- sysedge.mon file
 - adding entries
 - History Control table 511
 - Log Monitor table 506
 - Monitor table 494
 - NT Event Monitor table 508
 - Process Monitor table 499, 503
- backing store 493
- examples

- collecting disk transfer history 512
- monitoring 1-minute load average 497
- monitoring httpd process 505
- monitoring log files 507
- monitoring netscape process 502
- monitoring size of netscape process 502
- monitoring Windows event log files 510
- fields
 - History Control table 511
 - Log Monitor table 506
 - Monitor table 495
 - NT Event Monitor table 508
 - Process Monitor table 499, 503
- format 493
- history keyword 511
- installation directory 86
- location 86
- logmon keyword 506
- monitor keyword 494
- nventmon keyword 508
- processmon keyword 499
- procgrouppmon keyword 503
- sample entries
 - History Control table 512
 - Log Monitor table 507
 - Monitor table 497
 - NT Event Monitor table 510
 - Process Group Monitor table 505
 - Process Monitor table 501
- updating 433
- sysedge.mon.5 man page 91
- syslog facility
 - configuring alternative UNIX 134
 - Windows 134
- facility codes 490
- logging
 - agent-operation messages 155
 - daemon messages 491
 - messages 489
 - overview 489
 - priority codes 491
- SystemEDGE agent
 - configuration
 - files 86
 - overview 117
 - configuring support for debugging 135
 - current information 26
 - default settings 130
 - error messages 439
 - guidelines 43
 - installation directories 90
 - licensing overview 105
 - logging messages 155
 - overview 27
 - removing 98
 - self-monitoring features 31
 - specifying actions 35
 - starting 149
 - supporting custom MIB objects 34
 - traps 215
 - troubleshooting 423
 - uninstalling 98
 - using with
 - eHealth 40
 - Fault Manager 42
 - Windows
 - event monitoring 33
 - extensions 34
- Systems Management MIB
 - adding support for Windows registry and

- performance counters 403
- boot configuration parameters 169
- configuring support
 - Group information 133
 - history collection 142
 - Log Monitor table 141
 - NT Event Monitor table 141
 - Process Group Monitor table 141
 - Process Monitor table 139
 - Remote Shell group 133
 - User information 133
 - Who Table 132
- CPU Statistics group 211
- Disk Statistics group 208
- extending monitoring capability 391
- Extension group
 - overview 391
 - using 212
- History table 205
- Host System group 166
- I/O buffer cache 184
- interprocess communications 180
- kernel
 - configuration parameters 168
 - performance 178
- Log Monitor table 204
- message buffers 182
- Monitor table 202
- mounted devices 167
- NFS group 186
- NT
 - Cache Performance group 194
 - Event Monitor group 197
 - Memory Performance group 195
 - Page File Performance group 197
 - Registry and Performance Extension group 199
 - Registry group 191
 - Service group 192
 - System group 188
 - System Performance group 193
 - Thread group 190
- ntRegPerf group 403
- overview 29, 165
- Process Monitor table 203
- process table 174
- remote command execution 177
- RPC group 185
- self-monitoring tables 30
- streams
 - buffers 183
 - group 170
- system information 166
- unsupported MIB objects on Windows 199
- user
 - accounts 172
 - groups 173
 - logged on to system 176
- Who Table 176
- Windows-specific groups 188
- sysvariable utility
 - directory 91
 - error messages 484
 - using 426
- sysvariable.exe 93
- sysvariable.txt 94

T

threshold monitoring

examples 264

introduction 31

sample entry 246

Top Processes AIM

configuring support 144

overview 33

tracking assets 34

trap communities

configuring 130

overview 50

Trap PDU format 215

traps

address changing 228

authentication failure 132

changing process group 229

enabling monitorClear traps 254

format 230

implementing severity levels 436

license 227

LogMonMatch 218

LogMonNotReady 219

monitorClear

description 222

enabling 260

setting in Monitor table 254

monitorEntryNotReady 217, 254

monitorEvent 216

ntEventMonMatch 220

ntEventMonNotReady 221

private enterprise 215

processClear

description 226

enabling in Process Monitor table
293

processStart

description 224

disabling in Process Monitor table
293

processStop

description 223

setting in Process Monitor table 292

processThreshold 225

supported by SystemEDGE agent 215

troubleshooting

agent not responding to SNMP requests
425

agent not running 430

diagsysedge.exe 423

error messages 439

failed bind call 433

licensing 429

management system not receiving SNMP
traps 428obtaining diagnostic information about
the agent 423

required system patches 437

restarting processes 436

SNMP traps 436

SystemEDGE agent 423

tool 423

trap messages 436

updating sysedge.mon 433

verifying that the agent is running 423,
426

Tru64

installing SystemEDGE 84

platform support 23

removing SystemEDGE 103

SNMP agent 163

system requirements 84

U

- uninstalling SystemEDGE
 - AIX 102
 - Digital UNIX 103
 - HP-UX 101
 - Linux 102
 - Solaris 98
 - Tru64 103
 - Windows 98
- unmounting devices 168
- unsupported Windows performance data types 407
- updating sysedge.mon file 433
- User table, Systems Management MIB 172
- using
 - edgemon utility 274
 - edgemon utility 345
 - eHealth application insight modules 39
 - eHealth Service Availability 37
 - emphistory utility 387
 - extension variables with management software 399
 - Live Health 42
 - ntRegPerf variables with management software 410
 - performance registry variables 405
 - sysvariable utility 426
 - watch procgroup directive 329
- utilities
 - edgemon
 - directory 90
 - using 274
 - edgemon.exe 93
 - edgemon
 - directory 90
 - monitoring log files 345
 - monitoring processes 300
 - process monitoring 310

- edgemon.exe 93
- emphistory 90
- emphistory.exe 93
- license 109
- nteventmon error messages 479
- restartsvc.exe 93
- sendtrap 90
- sendtrap.exe 92
- snmpget error messages 483
- sysvariable 91
- sysvariable.exe 93
- walktree error messages 485
- xtrapmon 91
- xtrapmon.exe 93

V

- verifying
 - agent
 - response to queries 426
 - starting at system initialization 426
 - status 426
 - that the agent is running 423, 426

W

- walktree utility, error messages 485
- walktree.exe 93
- walktree.txt
 - UNIX 91
 - Windows 94
- watch logfile directive
 - examples 344
 - parameters 344
 - syntax 343
- watch ntevent directive
 - examples 371
 - parameters 370
 - syntax 370

watch procgroup directive

 examples 330

 parameters 329

 syntax 329

Who Table

 configuring support 132

 information 176

Windows

 adding registry variables 403

 configuring

 registry and performance variables
 408

 event monitoring

 examples 371

 introduction 33

 log-searching criteria 359

 overview 357

 Extensible agent 161

 installing SystemEDGE 59

 monitoring

 event logs 358

 registry and performance 405

 platform support 23

 registry extensions 34

 removing SystemEDGE 99

 starting Master agent 153

 stopping Master agent 150

 supported registry data types 405

 system requirements 59

 unsupported MIB objects 199

 writing extension scripts 397

X

xtrapmon utility

 description 91

 error messages 486

xtrapmon.1 man page 91

xtrapmon.exe 93

xtrapmon.txt 94